On Composing Generic Voting Schemes for Improved Privacy

Oskar Goldhahn[1]

NTNU: Norwegian University of Science and Technology, Trondheim Norway
oskar.goldhahn@ntnu.no

Abstract. Hybrid encryption provides a way for schemes to distribute trust among many computational assumptions, for instance by composing existing schemes. This is increasingly relevant as quantum computing advances because it lets us get the best of both worlds from the privacy of the post quantum schemes and the more battle tested classical schemes. We show how to compose members of a very general class of voting schemes and prove that this preserves correctness and integrity and improves privacy compared to its constituent parts. We also show an example composition using a lattice based decryption mixnet where the improvement in privacy can indirectly lead to an improvement in integrity.

1 Introduction

Due to the looming threat of quantum computing on cryptography there is ongoing work to replace cryptographic protocols and primitives with ones that are resistant against attacks by quantum computers. For many applications this is urgent because data needs to stay private long into the future and attackers may store encrypted data to extract information later once they have the capability. E-voting is one such application. Votes should ideally be kept private far into the future, though integrity only needs to hold until the election result is accepted.

Post-quantum voting is not without challenges. Despite the widespread study of post-quantum cryptography some promising encryption schemes have turned out to be insecure in the past [7, 2], even against classical attacks. Switching to quantum-safe cryptography is therefore not without risks. Making efficient, flexible and secure post-quantum schemes is also challenging. Practical post-quantum schemes have large proofs [1], require votes to have low entropy [5, 4] or sacrifice some security [6].

One way to guard against the risk associated with post-quantum cryptography is through the use of so-called hybrid cryptography, where we can design systems so that they remain secure as long as one of the (several) underlying computational problems remain hard. We propose using hybrid voting schemes to harden post-quantum voting schemes against attacks on their post-quantum assumptions, and also make some post-quantum schemes more practical by improving their security in a world where cryptographically relevant quantum computers do not exist.

1.1 Related Work and Alternative Approaches to Hybridization

To our knowledge the idea of hybridizing voting schemes to retain privacy when primitives are broken is absent from the literature, though there have been other attempts at solving the problem.

Many attempts aim for everlasting privacy where rather than distributing trust over many assumptions one seeks to avoid the need for any computational assumptions whatsoever for privacy after the tally. There are two main problems with everlasting privacy in practice. Firstly, most schemes need strong non-computational assumptions, such as trustees deleting their data after the election, or voters having access to an anonymous channel to cast their ballots [8]. Secondly, schemes need to be designed from the bottom up with everlasting privacy in mind, which limits the available set of schemes.

Public key cryptography has standard methods for building hybrid encryption from multiple public key cryptosystems [3]. One such method is to use secret sharing to split the message, encrypt the components in parallel and recombine the decryptions to decrypt. This does not work as easily for voting schemes because voting schemes need messages to be valid votes, and filter out those that are not. Checking validity of votes that are distributed between multiple different ciphertexts using different encryption schemes requires heavy weight zero knowledge proofs. This strategy can work for voting schemes where tally results of the different buckets of shared votes can be recombined, such as with additively homomorphic voting schemes. It seems very difficult to make this work with any shuffle based voting scheme.

For voting schemes that only use swappable cryptographic primitives that do not tightly integrate with each other it is also possible to hybridize the primitives instead of hybridizing the voting scheme itself. Schemes that use zero knowledge proofs are incompatible with this approach when the zero knowledge proofs are specialized and relate to one of the other primitives, such as encryption. One might be able to get away with only hybridizing some of the primitives, but this is voting scheme specific and requires additional analysis for each scheme. In practice it also requires the primitives in use to have sufficiently efficient hybrid alternatives, which poses a problem for generic zero knowledge proofs.

The approach of hybridizing primitives is compatible with some non-verifiable decryption mixnets [6], which only use encryption and signatures. Despite this we will see in Section 7 that our solution can provide better integrity than a decryption mixnet with hybrid encryption in practice.

1.2 Our Contributions

We define a simple composition construction to build hybrid voting schemes using layered encryption and prove that the correctness follows from the correctness of its components, the integrity follows from the integrity of *both* its components, which only matters at the time of the election, and the post-tally privacy follows from the privacy of *either* of its components. The idea is to have an inner and an outer voting scheme (similar to onion routing schemes). The

voter casts their vote using the inner scheme to get a ballot, which is then cast again using the outer scheme to get a 'doubly encrypted' ballot. In the tallying process, the outer scheme is decrypted first, and then the inner is decrypted. The votes stay private even if the malicious trustees remember the keys they used during the tally, and we do not need an anonymous channel.

To make this composition work for a wide range of inner and outer schemes we also propose a very flexible definition for voting schemes, modelling tallying as a message passing process. Since the construction itself does not rely on any assumptions beyond those on the inner and outer scheme, one can simply swap out a voting scheme for another if new attacks arise. It is also possible to use a scheme with everlasting privacy as a component, making the composition inherit everlasting privacy, but also improve on privacy if the non-computational assumptions are broken. One can also compose more than two schemes by nesting the composition, though for the sake of conservative post-quantum making a hybrid of two schemes is usually seen as sufficient.

Our example construction uses some classically secure inner scheme (typically a dlog-based scheme using verifiable shuffles and distributed decryption) combined with an outer scheme based on a (quantum-safe) decryption mixnet. This example illustrates how the composition can give us efficient voting schemes with post-tally privacy at least as good as either of its constituent parts. Also, the example improves on the security of the decryption mixnet by making it unnecessary for voters to completely reveal their vote to the public to prove tampering happened in the decryption mixnet when all trustees are malicious.

1.3 Challenges

At first glance our definitions and approach may seem needlessly complex. We have considered multiple simpler alternatives that did not work out or were deemed less desirable for various reasons, and we have favored flexibility over simplicity in the approach we chose.

Much of the complexity of our approach, especially in the definitions, comes from handling overlap between the two tallies in the composition. It is possible to avoid this complexity by determining a separation point, either before or during the tally, after which continuing with the outer tally is prohibited and starting with the inner tally is permitted. One may for example designate two hours for the tally and put the separation point in the middle.

One problem with this is that one needs to choose between a conservative separation point where a lot of the tally is spent waiting and a risky separation point where tallies can fail from being too slow, possibly due to an attacker.

Alternatively, agreeing on a separation point during the tally requires adding signatures with additional assumptions, interacts badly with threshold tallying and falls victim to the Two Generals' Problem without additional assumptions.

A separation point also loses some efficiency and flexibility because some trustees that could otherwise continue doing work will have to wait for the separation point.

2 Networks

We will need to create a tally procedure that is a composition of two distinct tally procedures. There are two main obstacles. The first is that we want to support very different structures on the tally procedures with one compositional design. The second is that the execution of the two tally procedures may end up being interleaved.

The second obstacle is best illustrated by threshold schemes where we can obtain a valid result before all trustees are done with their part of the tally. This means that some or all of the trustees may proceed to the second tally before everyone has finished the first tally. It therefore becomes difficult to separate the two tallies in a composition without sacrificing the threshold functionality.

As noted, voting schemes are quite diverse. To make composition feasible without restricting to a small subset of schemes we provide a flexible definition that exposes the details of the tallying process by modelling it as message passing between multiple trustees orchestrated by a possibly malicious network. This should be sufficient to model most voting schemes that collect all ballots non-anonymously before tallying.

Since we wish to capture precisely which assumptions are needed for which properties we make minimal cryptographic assumptions on the network. There is no implicit encryption, and a malicious network can insert, delete, reorder or misdirect messages at will. If encryption or identification is needed for security, the voting scheme has to handle this by itself. The network does get to know the id of the sender, and the id of the intended receiver, which are either a trustee id or a special 'outside' id indicating messages intended for public consumption that will be used for audits and computing the result of the tally.

Messages can also be sent *from* the 'outside' id, which is used to give the list of ballots to be tallied to the trustees. This is needed because some schemes, such as the decryption mixnet in Section 7 [6], do not guarantee that there is a unique valid result, tally uniqueness, and there is a potential attack in the composition of schemes without tally uniqueness where the adversary can make trustees disagree on the ballots they are tallying. It should be noted that an adversary with full control over the ballots tallied can easily break privacy, which is why the privacy definition requires messages sent from the 'outside' id to be a superset of the submitted honest ballots. This is sufficiently weak for our composition results to hold, even for schemes without tally uniqueness.

Termination is defined implicitly by having the network return a list of messages rather than the next message to be processed. In the honest case this is the list of all messages sent to the special 'outside' recipient that is a standin for the outside world and any auditors.

Definition 1. *A* Network *is a set of three procedures modifying some state, as follows:*

- *An initialization procedure $\mathcal{I}$ taking some public data and setting up the state of the network.*

- *A send procedure $\mathcal{S}$ taking a sender id and a list of messages with recipient ids.*
- *A processing procedure $\mathcal{P}$ producing a message with one trustee id each for sender and receiver or producing a list of messages going outside.*

Since our tallying process is orchestrated by a network we need to define when networks are behaving honestly to get a correctness definition; when everyone is honest everything goes as expected.

The intuition says that an honest network processes exactly those messages that are sent, and with the correct recipient and sender. This works nicely for schemes where all trustees need to participate in the tally and we know exactly when we are done, but this is not the case for treshold schemes. In a treshold scheme only some trustees are needed to complete the tally, and messages can still be in flight even though we can compute a valid result. Because of this we want to be lenient and allow *semi-honest* networks to withhold or drop some messages. Instead we say that a network is *finished* if all the messages that should have been processed have been, which will be the *fully honest* case.

Definition 2. *A* semi-honest *network is one where for every sender, recipient pair the set of messages that have been sent between them since initialization are always a superset of the messages that have been processed between them since intitialization.*

A network is finished *if the sets of sent and processed messages are equal for every sender, recipient pair.*

We can define a trivial semi-honest network $\mathcal{HN}$ that is always finished when it processes a list. The internal state is a queue of messages with recipients, a set of sent message trustee pairs, and a set of outgoing messages. The initialization sets both sets to empty. The send procedure adds the input elements to the queue and or the list of outgoing messages depending on whether the recipient is a trustee or outside, given that it isn't present in the queue or set of sent pairs already, in which case we do nothing. The processing procedure outputs a message and trustee id from the queue and adds it to the set of sent pairs, or outputs the set of outgoing messages if the queue is empty.

3 Voting Schemes

When voting electronically, each voter encodes their intent into a *vote*, which can for example be a candidate id or an ordered list of candidate ids, depending on the electoral system used. With voting schemes we hide voter intent by making voters cast their votes into a *ballot* in a way which should hide the underlying vote. In our definition the ballot may also include some additional data, such as zero knowledge proofs. The ballots can then, by using a tally, be transformed into a *result* for the election.

Because the network returns a list of messages rather than a tally result we combine tally validation with computation of the result. In practice any concrete voting scheme is free to include the result in one or all of the messages and to

provide an additional algorithm to compute a tentative result that might be invalid.

We use two algorithms, $\mathcal{TI}$ and $\mathcal{TS}$ to perform the tally. The first takes the tally key and produces an initial state for the trustee, the second takes a message and potentially produces new ones. In the honest setting messages received are either the list of ballot-associated data pairs to be tallied, or an internal message sent between trustees.

Definition 3. *A* Voting Scheme *is a set of trustee ids, a set of votes, a set of ballots, a set of results, a* counting function *mapping multisets of votes to results, and 6 algorithms as follows:*

- *A key generation algorithm $\mathcal{KG}$ producing a casting key, a validation key, and a list of tallying keys, one for each trustee.*
- *A casting algorithm $\mathcal{C}$ taking a casting key, a vote and associated data and producing a ballot.*
- *A tally initialization algorithm $\mathcal{TI}$ taking a tallying key and producing a new trustee state.*
- *A tally step algorithm $\mathcal{TS}$ taking a message, the id of the sender, and a tallier state and producing a new tallier state and a list of messages with recipient ids.*
- *A tally validation algorithm $\mathcal{V}$ taking a validation key, a list of ballot-associated data pairs and a list of messages and producing a result or $\perp$*

Adversaries interact with the tally as a dishonest network, which intercepts every message going through the network and may produce messages that were never sent by honest trustees.

The associated data is tied to a unique voter and used to prevent copy-attacks; making sure that one can't take an honestly generated ballot from one voter and repurpose it for another. Doing so would allow an adversary to skew the result in a way that depends on hidden votes. The details of the associated data depends on the system responsible for voter eligibility, which is beyond the scope of this paper, but since hashing is always a possibility we will assume it is a 256-bit value.

Now that we have the networks, voting schemes and tally adversary in place we can define how the tally itself proceeds. To simplify reasoning we use sequential computation and assume that trustees only do computation after they receive a message. In practice they may want to send one message at a time as they are computed rather than making all the progress that is possible without further messages before sending them all at once. Of course, the computation can be made asychronous without compromising security, for example by having the trustees use a message queue internally and only handling a new message once no more progress can be made without handling a new message.

As mentioned in Section 2, we want to allow adversaries to have some control over which ballots each trustee tries to tally, so we send the ballots over the network instead of giving them to the trustees right away.

The tally starts by initializing the state of the trustees, continues by sending the ballots to be tallied to each trustee over the network and finally hands messages to trustees one at a time until the network decides that we are done.

We end the tally once the network decides to send a list of outgoing messages. Semihonest networks have to do this once they run out of messages. The list of outgoing messages can then be used with $\mathcal{VS}_{\mathcal{V}}$ to try producing a result.

Definition 4. *The* Tally *procedure takes a casting key* ck*, a validation key* vk*, a list of tally keys,* $\overrightarrow{\mathsf{sk}}$*, a list of trustee ids indicating honest parties,* α *and a list of ballot-associated data pairs. It uses a voting scheme,* $\mathcal{VS}$*, and interacts with a network,* $\mathcal{N}$*. It goes through the following steps:*

1. *Initialize an empty list of processed messages with 'outside' as sender.*
2. *Initialize the network by giving it* ck *and* vk*, and use* $\mathcal{N}_{\mathcal{S}}$ *to send the list of ballot-associated data pairs to each of the honest trustees.*
3. *Use* $\mathcal{VS}_{\mathcal{TI}}$ *with the tally keys to initialize the state of each honest trustee.*
4. *Use* $\mathcal{N}_{\mathcal{P}}$*. If the obtained value is a list return it and the list of processed messages with 'outside' as sender. Otherwise the value is a message with sender and recipient.*
5. *If the sender is 'outside', add the message and recipient to the list of processed messages with 'outside' as sender.*
6. *If the recipient is an honest trustee, use* $\mathcal{VS}_{\mathcal{TS}}$ *to give them the message, modify the state and send the returned list of messages.*
7. *Go back to step 4.*

As mentioned, to support threshold schemes in the composition we need the tally to proceed even if some trustees do not participate. This means that schemes that may give the wrong result when messages are dropped cannot be used in a correct composition. To address this we require that when using semi-honest networks the tally does not produce a wrong result in addition to it producing the right result when the network is finished. If the network is unfinished the output of the tally may not validate, which means there is no result.

We also allow networks with procedures that may not terminate because this simplifies reductions. It also means that the tally may not terminate because the network fails to advance. For correctness we require that this is the only case where the tally does not terminate.

Rather than reasoning about unconditional correctness, we allow a small chance of failure to allow schemes that defend against copy attacks by removing duplicates. Removing duplicates turns out to be the only way we can defend against copy attacks on the internal ballot in the composition. Usually one would defend against copy-attacks by using the associated data, but for composition to have privacy even when the internal voting scheme does not there cannot be any information in the internal ballots that could identify the voters. The internal ballots use random associated data instead, and the tally de-duplicates colliding internal ballot-associated data pairs.

Definition 5. *Given a list of votes* $\overrightarrow{v}$*, a voting scheme,* $\mathcal{VS}$*, with counting function* f *is* $(\overrightarrow{v}, \overrightarrow{\text{ad}})$-correct *if for any semi-honest network* $\mathcal{N}$*;*

1. *With overwhelming probability, when we generate keys with* $\mathcal{KG}_{\mathcal{VS}}$*, cast all the votes in* $\overrightarrow{v}$ *with the resulting casting key and* $\mathcal{C}_{\mathcal{VS}}$ *and the associated data from* $\overrightarrow{\text{ad}}$*. When we complete a fully honest tally with those ballots we get a list of messages, and using* $\mathcal{V}$ *on those messages either produces* $\perp$ *or* $f(\overrightarrow{v})$*. Additionally, if* $\mathcal{N}$ *was finished* $\mathcal{V}$ *does not return* $\perp$*.*
2. *A fully honest tally with any inputs either terminates or gets stuck in one of the* $\mathcal{N}$ *procedures with overwhelming probability.*

A voting scheme is m-correct *if it is correct for all lists of at most* m *vote associated data pairs. It is simply* correct *if it is correct for all lists.*

4 Composition

If we have one arbitrary scheme, $\mathcal{VS}^{\mathsf{i}}$, and another, $\mathcal{VS}^{\mathsf{o}}$ whose counting function is sorting and we can encode ballot associated data pairs from $\mathcal{VS}^{\mathsf{i}}$ into votes of $\mathcal{VS}_1$ we can compose the two schemes by casting votes in a nested way using the casting algorithm of both schemes and tallying by removing one layer after the other, yielding a new scheme $\mathcal{VS}$ whose counting function is that of $\mathcal{VS}^{\mathsf{i}}$. One can think of $\mathcal{VS}^{\mathsf{i}}$ as the *inner* or *final* scheme and $\mathcal{VS}^{\mathsf{o}}$ as the *outer* or *initial* scheme, depending on whether one is looking at the structure of the ballots or the tallying procedure.

We separate messages for the two schemes by tagging them. The trustee-sent messages for $\mathcal{VS}^{\mathsf{i}}$ and $\mathcal{VS}^{\mathsf{o}}$ are tagged with 1 and 2 respectively. We also have additional messages tagged 1.5, which are copies of the outgoing messages of $\mathcal{VS}^{\mathsf{o}}$ instead sent to the other trustees to make it possible for them to compute the result of the initial scheme. The tags are only important for correctness and are not encrypted.

Intuitively having two layers of encryption on the ballots rather than one should provide some additional security, and this can be done iteratively to further increase it.

The restriction that $\mathcal{VS}^{\mathsf{o}}$ should compute sorting means that in practice this currently has to be a mixnet. For the post-quantum case this means that one might want $\mathcal{VS}^{\mathsf{o}}$ to be the classical one, allowing more flexibility for the post-quantum voting scheme.

Definition 6. *The* composition *of two voting schemes* $\mathcal{VS}^{o}$ *and* $\mathcal{VS}^{i}$ *with the same trustee ids where* $\mathcal{VS}^{o}$*'s counting function is sorting and its consistency check is independent of the ordering of the votes, where* $\mathcal{VS}^{i}$*'s tally step, validation and consistency check are independent from the ordering of the ballots and where the set of ballot associated data pairs of* $\mathcal{VS}^{i}$ *is a subset of the votes of* $\mathcal{VS}^{o}$ *is a voting scheme with the same counting function as* $\mathcal{VS}^{i}$ *defined as follows;*

- *The key generation runs* $\mathcal{VS}^{o}_{\mathcal{KG}}$ *and* $\mathcal{VS}^{i}_{\mathcal{KG}}$ *seqentially and returns the keys as pairs. We also add both validation keys to all the trustee keys.*

– *The casting algorithm runs $\mathcal{VS}^i_{\mathcal{C}}$ on the vote with the associated data being sampled uniformly at random, and casts the resulting pair as a $\mathcal{VS}^o$ vote with $\mathcal{VS}^o_{\mathcal{C}}$ using the real associated data, yielding the final ballot.*
– *The tally initialization algorithm runs the tally initialization for $\mathcal{VS}^o$ and $\mathcal{VS}^i$. The initialized state is the pair of states returned by the two tally initialization algorithms, the validation key for $\mathcal{VS}^o$, an empty 1.5 message list used for $\mathcal{VS}^o$ messages to 'outside' and a 2-initialized state tag indicating whether we are still waiting for a result for $\mathcal{VS}^o$ or have properly started tallying $\mathcal{VS}^i$.*
– *For the tally step algorithm any message received is first added to the message queue. There are five cases;*
  - *If we receive a message from 'outside', we use $\mathcal{VS}^o_{\mathcal{TS}}$ on it. We tag the returned messages to other trustees with* 1. *Messages to 'outside' get a 1.5 tag and are also sent to all the other trustees. We use the new $\mathcal{VS}^o$ state to modify the state of the composition and add the messages tagged with 1.5 to the 1.5 message list in the state. We return the new state and the list of tagged messages.*
  - *If we receive a message tagged 2 from a trustee we use $\mathcal{VS}^o_{\mathcal{TS}}$ on the message with the tag removed and the $\mathcal{VS}^o$ trustee state to produce the new $\mathcal{VS}^o$ state and a list of messages. For the messages we tag and add additional messages as described in the previous case. We return the modified trustee state and message list.*
  - *If our state tag is 2 and we receive a message tagged 1.5 from a trustee we add it to the 1.5 message list with the tag removed and try to use $\mathcal{VS}^o_{\mathcal{V}}$ to validate this list of messages and obtain a result for $\mathcal{VS}^o$. If it validates with a result we take the ballot associated data pairs in the result as a message with sender 'outside' and feed it into $\mathcal{VS}^i_{\mathcal{TS}}$ together with the $\mathcal{VS}^i$ state to produce a new $\mathcal{VS}^i$ state and list of messages, which we tag with 1. The output state has the new $\mathcal{VS}^i$ state and a 2 state tag. The output messages are those we tagged with 2. If the result does not validate we return the state with the modified 1.5 message list and an empty message list.*
  - *If we receive a message tagged 1 from a trustee we use $\mathcal{VS}^i_{\mathcal{TS}}$ on the message with the tag removed and the $\mathcal{VS}^i$ trustee state to produce the new $\mathcal{VS}^i$ state and a list of messages. We tag all the messages with 1 and return the modified trustee state and message list.*
  - *In any other case we return the old trustee state and an empty message list.*
– *The tally validation algorithm starts by using the tally validation of $\mathcal{VS}^o$ on the messages tagged 2. If this produces a result we use $\mathcal{VS}^i_{\mathcal{V}}$ on the resulting list of ballots and the messages tagged $\mathcal{VS}^i$ and return its output. If the tally validation of $\mathcal{VS}^o$ fails (returns $\perp$) we return $\perp$.*

Theorem 1. *If $\mathcal{VS}^o$ is m-correct, $\mathcal{VS}^i$ is m-correct, and they are composable, then their composition, $\mathcal{VS}$, is m-correct.*

Proof. We permit ourselves to be somewhat imprecise and treat overwhelming probability as certain to clarify the exposition. Adding more precise probability bounds is simple. We also don't construct any reductions explicitly.

Consider the tally for a given semi-honest network, $\mathcal{N}$, in the correctness game. From the correctness of $\mathcal{VS}^{\circ}$ every trustee with state tag 1 must have gotten a list of 1.5 messages yielding the correct result for the initial scheme. It follows, using the correctness of $\mathcal{VS}^{\mathsf{i}}$, that if the tally terminates then the messages validate to the correct result, or $\perp$.

Suppose $\mathcal{N}$ is finished when the tally is done. Then all messages tagged 2 and 1.5 must have been received, which from the correctness of $\mathcal{VS}^{\circ}$ means that every trustee must have been able to compute the correct list of ballots associated data pairs for $\mathcal{VS}^{\mathsf{i}}$. Due to the randomized associated data there are no duplicates. Since $\mathcal{N}$ is finished all messages marked 1 must also have reached their destination, which from the correctness of $\mathcal{VS}^{\mathsf{i}}$ means that the tally produces a valid result.

As for termination; a non-terminating tally with a semi-honest network, $\mathcal{N}$ either gets stuck in one of the $\mathcal{N}$ procedures, one of the $\mathcal{VS}$ procedures, or an unbounded number of messages must have been sent. The correctness of $\mathcal{VS}^{\circ}$ and $\mathcal{VS}^{\mathsf{i}}$ eliminate the last two options, so if we get stuck this must be because the network fails to proceed. □

5 Integrity

Integrity informally means that we don't get the wrong result under adversarial conditions. There are multiple ways to make this formal depending on what one considers to be a 'wrong' result. The strictest version is that every ballot, even the adversarially generated, have an underlying vote, and performing a tally that validates gives the same result as just counting those votes directly. This definition is too strict for us for two reasons. Firstly, this requires every ballot to be valid, though this can be worked around by having an explicit blank vote that doesn't affect the tally. Secondly, it does not permit schemes where the result of a valid tally is not uniquely determined by the ballots, such as decryption mixnets, where ballots cast by a voter conspiring with a trustee can be substituted by said trustee during the tally without detection. Instead we only require there to be a sensible set of votes that could have produced the result after the fact and that this set contains all the honest votes.

The above discussion indicates that we want to distinguish between adversarially and honestly generated ballots somehow. We do this by making an adversary submit ballots through a Ballot Box oracle, either as adversarially generated, or as honestly generated from an adversarially chosen vote.

Definition 7. *An* integrity adversary *is a stateful actor that can act as a network and has an additional procedure $\mathcal{FBB}$ taking a public key and a list of private keys, and producing a list of ballot associated data pairs. The adversary has access to a stateful* Ballot Box *oracle with the following procedures;*

– *An initialization procedure taking a public key, storing it and initializing an empty list.*
– *A cast procedure taking a vote and associated data and producing a ballot using the public key, adding all of the associated data, vote and ballot to the list and returning the ballot.*
– *A read procedure that produces the list.*

The first and the last are hidden from the adversary.

A standard, though we frame it unconventionally, way to define integrity without the need for tally uniqueness is by having some algorithm that can look at the result, the honest votes, and the number of ballots and determine whether all the honest votes were taken into account for the result and whether the total number of votes taken into account is bounded by the number of ballots. Let's call this a 'consistency check' because it checks whether the result is consistent with the honest votes. Since this is part of the game we can give the algorithm additional information about the tally, such as secret keys and the ballots used in the tally. For counting functions that just sort or count the votes a consistency check can perform the tally on the honest votes and then use the subset relation and cardinality for multisets to check that the honest votes are in the computed result and that the number of votes is correct. For many voting schemes the consistency check can work by decrypting the ballots and tallying the plaintext votes, though this is too strong for schemes lacking tally uniqueness. Since we want to support a wide variety of voting schemes we leave the consistency check abstract. Because of this we need a different definition than is common in the literature, which usually has a specific consistency check, either implicitly or explicitly.

It should be noted that this definition may still give voters conspiring with trustees one advantage over others. They might be able to look at the result or other votes before deciding their own vote. This is a problem with *any* scheme, but for schemes with tally uniqueness this generally only works if the conspiring trustees are many enough to compute tallies. For schemes without tally uniqueness the attack can happen during the real tally, but this can be mitigated ad hoc by hiding the election result until it is uniquely determined. The lattice based decryption mixnet by Boyen, Haines and Müller [1] does this. Mitigations for schemes with tally uniqueness include hiding the keys from the trustees until all the ballots have been cast. We do not try to capture this attack with our definitions, and instead leave the concern for specific instantiations.

The goal of the adversary is to produce an inconsistent tally result, so using a conservative consistency check that has some false negatives is fine. It only makes the adversary more likely to win the game. The advantage is that a conservative consistency check may be more efficiently computable, which could make it easier to use integrity in reductions.

Definition 8. *A* consistency check *for a voting scheme is a function taking a casting key, a validation key, a list of private keys, a multiset of ballot associated data pairs* $\mathcal{A}_i = (b, \mathrm{ad})_i$*, a multiset of votes with ballot associated data pairs*

$\mathcal{H}_i = (v, b', \mathrm{ad}')_i$, *and a list of messages,* $\overrightarrow{m}$*, producing a boolean such that; if* $\mathcal{V}(\overrightarrow{m})$ *gives a valid result* r*, then the consistency check producing true implies the existence of a multiset of votes that is a superset of those in* $\mathcal{H}$ *with cardinality bounded by* $\#\mathcal{H} + \#\mathcal{A}$*, such that counting them produces* r*. Meaning that there exists an assignment of votes of a subset of the adversarial ballots such that counting them together with the honest votes produces the result.*

After the preceding discussion the integrity game follows naturally; we allow the adversary to generate ballots, perform the tally, and check consistency at the end.

Definition 9. *The* α*-integrity game for a set of honest trustees* α*, a voting scheme* $\mathcal{VS}$ *and an integrity adversary,* $\mathcal{IA}$*, is defined as follows;*

- *Initialize* $\mathcal{VS}$ *and collect the trustee keys for the dishonest parties in a list.*
- *Initialize the ballot box with the public key.*
- *Use* $\mathcal{IA}_{\mathcal{FBB}}$ *with the keys for the dishonest parties and the public key to let the adversary fill the ballot box and provide a list of ballots with associated data.*
- *Read the contents of the ballot box and use it to add information about the underlying votes to the adversarially provided ballots for those that match.*
- *Use the keys to perform* Tally *with the ballot associated data pairs from the adversary,* α *as the honest parties and* $\mathcal{IA}$ *as the network.*
- *Use* $\mathcal{VS}_{\mathcal{V}}$ *to validate the tally with the ballot associated data pairs from the adversary. If it does not, the adversary loses.*
- *Use the consistency check to check whether the tally output is consistent with the ballot associated data pairs provided by the adversary, the underlying votes for the honestly generated ballots from the ballot box and the keys. If it is not, the adversary wins. If the result is consistent, the adversary loses.*

The advantage of an adversary is just its probability of winning the game.

Because the adversary can replace messages it is not guaranteed that the trustees use the list of ballots we took from the Ballot Box. They may not even use the same list of ballots. If some form of synchronization is desired to make sure that the honest trustees are tallying the same list of ballots that is the responsibility of the voting scheme. It is strictly speaking not necessary for integrity. What is important, however, is that the auditors have some way of gaining confidence that the ballots they are checking were submitted by authorized voters in a compliant way and that honestly generated ballots were not prevented from being submitted, but this is orthogonal to the concerns we address.

5.1 Composition integrity

This is the point where it becomes important that the adversary has some control over the ballots that are tallied. The trustees might transition from tallying $\mathcal{VS}^{\circ}$ to $\mathcal{VS}^{\mathsf{i}}$ at different times, so a lot of messages for $\mathcal{VS}^{\mathsf{i}}$ can be sent before another

trustee even knows which ballots to tally. This is fine in the fully honest setting because we are guaranteed that every party computes the same result from $\mathcal{VS}^{\circ}$ meaning we can just use the first result when simulating the interactions with $\mathcal{VS}^{\circ}$, but this need not be the case in the adversarial setting, even in settings where we have integrity for $\mathcal{VS}^{\circ}$. Consider a 2-threshold scheme with 3 trustees where adversarial votes can be swapped out during the tally. Then an adversarial trustee can use two different tallies with the other two parties where they replace different votes, yielding two different, but consistent, results.

Definition 10. *We define a consistency check for the composition that performs honest $\mathcal{VS}^{\circ}$ single ballot tally to get the inner ballots for the honest ones and then uses the consistency checks for both $\mathcal{VS}^{\circ}$ and $\mathcal{VS}^{i}$. If the single ballot tallies fail we return false.*

Theorem 2. *If $\mathcal{VS}^{\circ}$ is 1-correct, and $\mathcal{VS}^{\circ}$ and $\mathcal{VS}^{i}$ have α-integrity, then the composition if they are composable, $\mathcal{VS}$, has α-integrity.*

Proof. We again permit ourselves to be somewhat imprecise and treat overwhelming probability as certain to clarify the exposition. Adding more precise probability bounds is simple.

If an adversary, $\mathcal{A}$ wins the α-integrity game, then the consistency check must have failed, but all the others must have passed. Since $\mathcal{VS}^{\circ}$ is 1-correct the internal ballots used in the consistency check are correct. The consistency check for either $\mathcal{VS}^{\circ}$ or $\mathcal{VS}^{\mathsf{i}}$ must have failed, which means that for the two adversaries $\mathcal{A}^{\circ}$ and $\mathcal{A}^{\mathsf{i}}$ against the α-integrity game of $\mathcal{VS}^{\circ}$ and $\mathcal{VS}^{\mathsf{i}}$ respectively that use $\mathcal{A}$ and simulate the interactions with the other voting scheme, one of the two must win. Therefore we can bound the advantage of $\mathcal{A}$ by the sum of the advantages of $\mathcal{A}^{\circ}$ and $\mathcal{A}^{\mathsf{i}}$. □

6 Privacy

We frame privacy as a distinguishing game between two sets of honestly generated ballots. As in integrity the adversary generates them by using a Ballot Box oracle, though in this case the Ballot Box maintains two lists of ballots, only one of which is revealed to the adversary.

Usually one would require the adversary to provide pairs of votes, one for each side of the distinguishing game, but this is not sufficient for composition because the outer layer may produce distinct ballots for identical votes. To address this we instead allow the adversary to provide pairs of randomized algorithms producing the votes. At the end the game checks that for the indices of the ballots used, the algorithms used to generate them are just a permutation of the algorithms on the other side. It is easy to see that this is stronger than the usual game since that is equivalent to only permitting deterministic algorithms, but the other direction is not so easy. We do have both reductions if we restrict the adversary to have to use all generated ballots, meaning only static attacks against the voters. However, this is not so realistic and fails to capture some

attacks where the adversary is looking for some kind of collision in the ballots and makes voters re-cast their votes until such a collision is found, or where the adversary can learn information about ballots that are not being counted. Morally, allowing randomized vote generation should not strengthen the adversary since using randomly generated votes only reduces the information available to the adversary. We believe that existing schemes can have their privacy proofs modified to work in the randomized setting.

Definition 11. *A* privacy adversary *is a stateful actor that can act as a network and has two additional procedures, a guessing procedure, $\mathcal{G}$, taking no input and returning a bit, and a ballot box filling procedure, $\mathcal{FBB}$, taking a public key and a list of private keys and producing a list of ballots. The adversary has access to a stateful* Double Ballot Box *oracle, with the following procedures;*

- *An initialization procedure taking a bit, b, and a public key, storing both and initializing two empty lists.*
- *A cast procedure taking associated data and two randomized vote producing algorithms. The cast procedure produces ballots from the randomized algorithms by sampling a vote from the algorithms, using the public key and adding the associated data, first algorithm and ballot to the first list and the associated data, second algorithm and ballot to the second. It returns the inserted ballot corresponding to b.*
- *A read procedure that produces the lists.*

Only the second procedure is accessible to the adversary.

We use a somewhat weak privacy notion where we only require indistinguishability between pairs of tallies where the honest votes are permuted. There are stronger notions that instead require indistinguishability between tallies yielding the same result, but this cannot in general be used for voting schemes that do not have tally uniqueness because adversarial trustees can replace adversarial votes during the tally, changing the result. A more fine grained analysis may be possible, but since deployed voting schemes are often mixnets which cannot support more than indistinguishability of permuted votes anyways we leave this as future work.

Definition 12. *The α-privacy game for a set of honest trustees α, a voting scheme $\mathcal{VS}$ and a privacy adversary, $\mathcal{PA}$, is defined as follows;*

- *Initialize $\mathcal{VS}$ and collect the trustee keys for the dishonest parties in a list.*
- *Flip a coin to select b and initialize the double ballot box with b and the public key.*
- *Use $\mathcal{PA}_{\mathcal{FBB}}$ with the keys for the dishonest parties and the public key to let the adversary cast the honest votes.*
- *Read the contents of the double ballot box.*
- *Check that for the intersection of the ballot associated data pairs provided by the adversary, and those in the b side of the ballot box, the list of algorithms corresponding to the $b-1$ side of the ballot box are just a permutation of those in the b side. If not, flip a coin to decide whether the adversary wins.*

- *Use the keys to perform* Tally *with the list of ballots provided by the adversary, with α as the honest parties and $\mathcal{PA}$ as the network.*
- *The tally reveals the processed messages with 'outside' as sender. We check that all these messages are lists of ballots containing the intersection of the ballots in the double ballot box list corresponding to b and those originally provided by the adversary, and that none of them contain any other ballots from the ballot box list. If not, flip a coin to decide whether the adversary wins.*
- *Use $\mathcal{PA}_{\mathcal{G}}$ to get the guess of the adversary. The adversary wins if the guess equals b, the bit used to select which ballots to tally.*

The advantage of the adversary is the difference between their winning and losing probability.

The list of ballots queried through the $\mathcal{C}$ procedure of the oracle are the *honest* votes, and the list of ballots provided by $\mathcal{PA}_{\mathcal{FBB}}$ are the *submitted* ballots, which need not be the same as those sent to the trustees, but we require that the trustees get all the submitted honest ballots, and no unsubmitted honest ballots.

In the real world an adversary could sneak in an honest ballot by copying one of the unsubmitted ones. This could cause issues in practice if an honest voter had their vote duplicated from an earlier ballot they failed to submit. Preventing this is the task of the associated data. The voting infrastructure should prevent submission of multiple ballots with the same associated data.

In the composition there is no such implicit voting infrastructure between the two voting schemes, so we have to handle this explicitly instead. The associated data of the internal ballots cannot be associated with any given voter, so even assigning unique associated data becomes difficult. This means that solving this requires a different solution, which we discuss in the end of the next section.

6.1 Composition Privacy

We reach our main contribution, the result that after the tally we only need to rely on the privacy of one scheme for the composition to have privacy. Note that privacy of the second scheme is not sufficient. We also need integrity of the first in that case. We also need privacy of the first scheme at the time of the tally to prevent copying of the internals of unsubmitted ballots, though there are ways to avoid this which we discuss at the end of this section. We discuss the implications of these conditions after the statement of theorem 4.

Theorem 3. *If $\mathcal{VS}^{\circ}$ has α-privacy then its composition with $\mathcal{VS}^{i}$, $\mathcal{VS}$ has α-privacy.*

Proof. Similar to the integrity game we use the adversary against the α-privacy game of the composition to build another, $\mathcal{A}^{\circ}$ against the α-privacy game of $\mathcal{VS}^{\circ}$ that simulates the interactions with $\mathcal{VS}^{\mathsf{i}}$. The resulting games are equivalent, so the result follows. □

Theorem 4. *Let* $\mathsf{Adv}^{\mathcal{VS}}_{Priv}\mathcal{A}$ *be the advantage of* $\mathcal{A}$ *against the* α*-privacy game of* $\mathcal{VS}$*, and* $\mathsf{Adv}^{\mathcal{VS}}_{Integ}\mathcal{A}$ *be the advantage of* $\mathcal{A}$ *against the* α*-integrity game of* $\mathcal{VS}$*, and* $\mathit{Coll}(q)$ *be the collision probability in a set of* q *uniformly sampled 256 bit values. Let* $\mathcal{VS}$ *be the composition of inner scheme* $\mathcal{VS}^i$ *and outer scheme* $\mathcal{VS}^o$*, with* t *trustees. Let* $\mathcal{A}$ *be an adversary against the* α*-privacy game of* $\mathcal{VS}$ *that submits or queries at most* q *ballots, then there exist adversaries* $\mathcal{A}^i$ *and* $\mathcal{A}^o$ *against the* α*-privacy games of* $\mathcal{VS}^i$ *and* $\mathcal{VS}^o$ *respectively and an adversary* $\mathcal{A}_{Integ}$ *against the* α*-integrity game of* $\mathcal{VS}^o$ *that all use* $\mathcal{A}$ *as an oracle, where* $\mathcal{A}^o$ *and* $\mathcal{A}_{Integ}$ *do not use* $\mathcal{A}_{\mathcal{G}}$ *such that*

$$\mathsf{Adv}^{\mathcal{VS}}_{Priv}(\mathcal{A}) \leq \mathsf{Adv}^{\mathcal{VS}}_{Priv}(\mathcal{A}^i) + \mathsf{Adv}^{\mathcal{VS}}_{Priv}(\mathcal{A}^o) + t \cdot \mathsf{Adv}^{\mathcal{VS}}_{Integ}(\mathcal{A}_{Integ}) + \mathit{Coll}(q)$$

The proof for this theorem can be found in appendix A.

The $\mathsf{Adv}^{\mathcal{VS}}_{\mathsf{Priv}}(\mathcal{A}^{\mathsf{o}})$ and $\mathsf{Adv}^{\mathcal{VS}}_{\mathsf{Integ}}(\mathcal{A}_{\mathsf{Integ}})$ terms detract somewhat from the result, but it is important to note that $\mathcal{A}^{\mathsf{o}}$ and $\mathcal{A}_{\mathsf{Integ}}$ do all the calls to $\mathcal{A}$ at the time of the election, and only to the parts of $\mathcal{A}$ that would need to be used during the election for $\mathcal{A}$ to mount an attack. This means that $\mathcal{A}$ being able to learn additional information about the votes in a current election in the future means that the current elections would need to be vulnerable to attacks today. Contrapositively this means that security in the present is sufficient for the $\mathsf{Adv}^{\mathcal{VS}}_{\mathsf{Priv}}(\mathcal{A}^{\mathsf{o}})$ and $\mathsf{Adv}^{\mathcal{VS}}_{\mathsf{Integ}}(\mathcal{A}_{\mathsf{Integ}})$ terms to vanish in practice.

We make some observations about the implications of this time dependence. Firstly, an attack that requires sophisticated techniques cannot be performed in the past before those techniques were developed, and the same also holds for technology such as quantum computers. Additionally, while many attacks against privacy can be performed against static data over long periods of time, attacking the privacy of the composition through the privacy or integrity of the first scheme requires the attacks to be performable in the time period between when the attack is discovered and the election is held, and any computation that requires the public keys or ballots need to wait until these are made available. Still, there is a possibility for an efficient attack against the first scheme that is kept hidden from the public for a long time to break the privacy of an election.

The $\mathsf{Adv}^{\mathcal{VS}}_{\mathsf{Integ}}(\mathcal{A}_{\mathsf{Integ}})$ term can be strengthened by using an inner scheme where integrity does not rely on computational assumptions.

As we will see in the proof, α-privacy of $\mathcal{VS}^{\mathsf{o}}$ is only needed to prevent the adversary from getting to the internals of unsubmitted ballots before the tally is finished, so if we can do this in a different manner we can have privacy of the composition even if the privacy of $\mathcal{VS}^{\mathsf{o}}$ fails at the time of the tally. One way of doing this is to add an external layer of threshold encryption to the ballots that becomes transparent after the tally. Since we only need to hide the unsubmitted ballots until after the tally we can even publish the secret keys of the threshold decryption after the tally, which means the auditors do not need to be sent any additional proofs for each voter. It is still necessary to check the decryption proofs however, both for the trustees before the rest of the tally and for the auditors on proofs they generate themselves using the secret keys. Depending on the threshold decryption scheme there may be a way to optimize the work

necessary for the prove and check steps of the auditors since they are done by the same party and they hold all the secret keys at that point.

Using a threshold decryption scheme allows us to switch out $\mathsf{Adv}^{\mathcal{VS}}_{\mathsf{Priv}}(\mathcal{A}^{\circ})$ in Theorem 4 for the advantage of some adversary against IND-CCA2 of the treshold decryption, but we can minimize the number of computational assumptions in this reduction by using the same computational assumption for IND-CCA2 of the treshold decryption as for the privacy of $\mathcal{VS}^{\mathsf{i}}$.

7 Example Construction

As an example instantiation of the composition, consider the post-quantum verifiable decryption mixnet with tripwires by Boyen, Haines and Müller [6]. Their scheme is efficient for a post-quantum scheme, can handle unstructured votes, and the counting function is sorting, which makes it eligible for the outer voting scheme in a composition. For the inner scheme in the composition we have ample choice between efficient classical voting schemes.

Note that we do not prove that the decryption mixnet, or any other voting scheme, satisfies our security notions. In fact, the decryption mixnet fails to be private against attacks copying the internals of unsubmitted ballots, though this can be addressed by adding an external layer of threshold decryption as described in the end of the previous section. For the example we entertain the idea that a similar scheme can be proven secure, and that some efficient classical voting scheme also can be.

The main weakness of the decryption mixnet is the fact that it does not have integrity in the fully adversarial setting when every part of the tally is tranparent to the trustees. This weakness can be mitigated by having voters check the tally, but proving misbehavior here requires revealing the vote, which might dissuade voters from doing so. This is also the case for a decryption mixnet using hybrid encryption. By using composition with a classical voting scheme the voters would only need to reveal the classical ballot, keeping their vote secret in the short term, which would lower this barrier.

The benefit against other post quantum schemes comes from efficiency since post-quantum schemes tend to have large proofs in the case of verifiable mixnets, or limitations on votes in the case of homomorphic voting.

Additionally, in a possible future where lattice cryptography is broken yet we still do not have quantum computers the composition would leave votes private, which using the decryption mixnet by itself would not have.

References

[1] Diego F. Aranha et al. “Verifiable Mix-Nets and Distributed Decryption for Voting from Lattice-Based Assumptions”. In: *ACM CCS 2023: 30th Conference on Computer and Communications Security*. Ed. by Weizhi Meng et al. Copenhagen, Denmark: ACM Press, Nov. 2023, pp. 1467–1481. DOI: 10.1145/3576915.3616683.

[2] Ward Beullens. "Breaking Rainbow Takes a Weekend on a Laptop". In: *Advances in Cryptology – CRYPTO 2022, Part II*. Ed. by Yevgeniy Dodis and Thomas Shrimpton. Vol. 13508. Lecture Notes in Computer Science. Santa Barbara, CA, USA: Springer, Cham, Switzerland, Aug. 2022, pp. 464–479. DOI: 10.1007/978-3-031-15979-4_16.

[3] Nina Bindel et al. "Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange". In: *Post-Quantum Cryptography - 10th International Conference, PQCrypto 2019*. Ed. by Jintai Ding and Rainer Steinwandt. Chongqing, China: Springer, Cham, Switzerland, May 2019, pp. 206–226. DOI: 10.1007/978-3-030-25510-7_12.

[4] Ian Black et al. *Practical Quantum-Safe Voting from Lattices, Extended*. Cryptology ePrint Archive, Report 2022/1686. 2022. URL: https://eprint.iacr.org/2022/1686.

[5] Xavier Boyen, Thomas Haines, and Johannes Mueller. *Epoque: Practical End-to-End Verifiable Post-Quantum-Secure E-Voting*. Cryptology ePrint Archive, Report 2021/304. 2021. URL: https://eprint.iacr.org/2021/304.

[6] Xavier Boyen, Thomas Haines, and Johannes Müller. "A Verifiable and Practical Lattice-Based Decryption Mix Net with External Auditing". In: *ESORICS 2020: 25th European Symposium on Research in Computer Security, Part II*. Ed. by Liqun Chen et al. Vol. 12309. Lecture Notes in Computer Science. Guildford, UK: Springer, Cham, Switzerland, Sept. 2020, pp. 336–356. DOI: 10.1007/978-3-030-59013-0_17.

[7] Wouter Castryck and Thomas Decru. "An Efficient Key Recovery Attack on SIDH". In: *Advances in Cryptology – EUROCRYPT 2023, Part V*. Ed. by Carmit Hazay and Martijn Stam. Vol. 14008. Lecture Notes in Computer Science. Lyon, France: Springer, Cham, Switzerland, Apr. 2023, pp. 423–447. DOI: 10.1007/978-3-031-30589-4_15.

[8] Thomas Haines et al. "SoK: Secure E-Voting with Everlasting Privacy". In: *Proceedings on Privacy Enhancing Technologies* 2023.1 (Jan. 2023), pp. 279–293. DOI: 10.56553/popets-2023-0017.

A Concrete Reduction

Definition 13. *Given two composable voting schemes $\mathcal{VS}^{\circ}$ and $\mathcal{VS}^{i}$, and an adversary $\mathcal{A}$ against the privacy game of the composition we define an adversary $\mathcal{A}^{i}$ against the privacy game of $\mathcal{VS}^{i}$ as follows;*

- *We use the oracle, $\mathcal{O}^{i}$ given to $\mathcal{A}^{i}$ to construct one, O for $\mathcal{A}$ as follows:*
  - *Initialization takes an $\mathcal{VS}^{\circ}$ casting key and collection of $\mathcal{VS}^{\circ}$ secret keys and initializes a list of queries*
  - *The cast procedure takes algorithms generating votes for $\mathcal{VS}^{i}$, casts them using $\mathcal{O}^{i}$ and adds another layer by generating random associated data and using $\mathcal{VS}^{\circ}_{\mathcal{C}}$, whose output is returned. Before returning, all the values used in this procedure are added to the list of queries.*
  - *The read procedure produces the list of queries.*

- *The ballot box filling procedure generates keys for $\mathcal{VS}^{\circ}$, initializes O with them and calls $\mathcal{A}_{\mathcal{FBB}}$, which yields a list of double layered ballots. Single ballot tallies for $\mathcal{VS}^{\circ}$ are used to strip off the outer layer, and the new list of single layered ballots is returned.*
- *The network initialization procedure runs the tally initialization of $\mathcal{VS}^{\circ}$ for each of the trustees and stores their states together with an empty list of 1.5 tagged messages for each trustee, and an empty list of $\mathcal{VS}^{i}$ ballots.*
- *The network sending procedure adds an 1 to messages not marked with 'outside' and feeds all messages to $\mathcal{A}_{\mathcal{S}}$*
- *The network processing procedure first uses $\mathcal{A}_{\mathcal{P}}$ to get a message and recipient. We proceed differently based on the tag of the message;*
  - *If the message is tagged with 1 we strip off the tag and return it.*
  - *If the tag is 1.5 we append it to the list of 1.5 messages for that trustee and try to validate it with $\mathcal{VS}^{i}_{\mathcal{V}}$. If it validates we return the list of ballots we get from the validation, with sender 'outside' and recipient being the trustee who got the 1.5 tagged message. We also add it to our list of $\mathcal{VS}^{i}$ ballots for the trustee. If it does not validate we return nothing.*
  - *If the message is tagged with 2 we strip off the tag and use the $\mathcal{VS}^{\circ}_{\mathcal{TS}}$ procedure to advance the state of the relevant $\mathcal{VS}^{\circ}$ trustee. The messages returned are passed through $\mathcal{A}$, with an added 2 tag if they are addressed to a trustee and an 1.5 tag if they are addressed to 'outside'. After sending it we go back to the start of this procedure and use $\mathcal{A}_{\mathcal{P}}$ again.*
- *The guessing procedure just uses the guessing procedure of $\mathcal{A}$*

$\mathcal{A}^{\mathsf{i}}$ behaves almost the same in the privacy game of $\mathcal{VS}^{\mathsf{i}}$ as $\mathcal{A}$ does for the game of $\mathcal{VS}$. The differences are only in the checks performed on the submitted ballots and ballots sent to the adversaries. In the $\mathcal{VS}^{\mathsf{i}}$ game we check the inner ballots, while in the $\mathcal{VS}$ game we check the outer ballots. For these checks to give the same result it is sufficient for the $\mathcal{VS}^{\circ}$ tally result to include the submitted honest votes and not add any additional unsubmitted honest votes.

Since we validate the tally results for each of the trustees the first condition amounts to the guarantee made by integrity. That is, if the tally validates all the honest votes were included. Since we need this to be true for all the trustees we get a loss equal to the number of trustees here.

The second condition is more complex. Intuitively it should be impossible to add unsubmitted votes without decrypting the ballots. Using this idea we can build an adversary against the privacy of $\mathcal{VS}^{\circ}$ that checks the ballots in the $\mathcal{VS}^{\circ}$ result against the unsubmitted $\mathcal{VS}^{\circ}$ ballots in the left and right game. If there is duplication with the left game we guess 0, if there is duplication with the right we guess 1, if there is duplication in neither or both we guess at random. Note that the probability of duplication with the game that is not being played is bounded by the collision probability of the random 256 bit associated data, so we expect duplication, if any, to just be in the game being played. If $\mathcal{VS}^{\circ}$ has privacy then the probability of such duplication must thus be negligible.

Of note is that the adversary against the privacy game of $\mathcal{VS}^{\circ}$ described above does not require using $\mathcal{A}_{\mathcal{G}}$, which means that the attack implied by the

reduction remains valid at the time of tally even if $\mathcal{A}_{\mathcal{G}}$ requires techniques or hardware that is not available at that time.

Theorem 4 follows from the above discussion.

Also note that instead of using the privacy of $\mathcal{VS}^{\circ}$ we could also modify the scheme to add a layer of encryption, and use IND-CCA2 of that. The only important part is that we can hide a challenge amongst the internals of the unsubmitted ballots. This modification would incur a loss proportional to the number of ballots because we would need to guess a ballot that stays unsubmitted to inject the challenge.