Dispute Resolution in Voting

David Basin
Department of Computer Science
ETH Zurich
basin@inf.ethz.ch

Saša Radomirović
Department of Computer Science
Heriot-Watt University
sasa.radomirovic@hw.ac.uk

Lara Schmid
Department of Computer Science
ETH Zurich
schmidla@inf.ethz.ch

***Abstract*—In voting, disputes arise when a voter claims that the voting authority is dishonest and did not correctly process his ballot while the authority claims to have followed the protocol. A dispute can be resolved if any third party can unambiguously determine who is right. We systematically characterize all relevant disputes for a generic, practically relevant, class of voting protocols. Based on our characterization, we propose a new definition of dispute resolution for voting that accounts for the possibility that both voters and the voting authority can make false claims and that voters may abstain from voting.**

A central aspect of our work is timeliness: a voter should possess the evidence required to resolve disputes no later than the election's end. We characterize what assumptions are necessary and sufficient for timeliness in terms of a communication topology for our voting protocol class. We formalize the dispute resolution properties and communication topologies symbolically. This provides the basis for verification of dispute resolution for a broad class of protocols. To demonstrate the utility of our model, we analyze a mixnet-based voting protocol and prove that it satisfies dispute resolution as well as verifiability and receipt-freeness. To prove our claims, we combine machine-checked proofs with traditional pen-and-paper proofs.

I. Introduction

For a society to accept a voting procedure, the public must believe that the system implementing it works as intended, that is, the system must be *trustworthy*. This is essential as elections involve participants from opposing political parties that may neither trust each other nor the election authority. Nevertheless, there must be a consensus on the final outcome, including whether the election is valid. This requires that voters and auditors can *verify* that the protocol proceeds as specified and detect any manipulations, even if they do not trust the authority running the election. To achieve this, the information relevant for checking verifiability may be published in a publicly accessible database, known as the *bulletin board*.

a) The need for dispute resolution: Ballots are cast privately in elections. Thus only the voters themselves know if and how they voted. If a voter claims that his ballot is incorrectly recorded or that he was hindered in recording his ballot, no other party can know, a priori, whether the voter is lying or if there was a problem for which the voting authority is responsible. We call such unresolved situations *disputes*.

When a dispute occurs, the honest parties must be protected. That is, an honest voter who detects some manipulation must be able to convince third parties that the authority was dishonest.[1] In particular, when a voter checks whether his cast ballot is correctly recorded, then either this is the case (respectively, no ballot is recorded when he abstained from voting) or he can convince others that the authority was dishonest. Another problem is when a voter cannot even proceed in the protocol to perform such checks, for instance when he is not provided with a necessary confirmation. Hence, a timeliness guarantee must ensure by the election's end that an honest voter's ballot is correctly recorded or there is evidence that proves to any third party that the authority is dishonest. Finally, in addition to protecting the honest voters from a dishonest authority, the honest authority must be protected from voters making false accusations. That is, when the authority is honest, no one should be able to convince others of the contrary.

b) State of the art: The vast majority of formal analyses of remote e-voting protocols do not consider dispute resolution at all, e.g., [1], [12], [15], [16]. Works that recognize the importance of dispute resolution [6], [2] or that take aspects of it into account when proposing poll-site [7], [10], [11], [13], [17], [24], [38] or remote [39] voting protocols, reason about it only informally. The most closely related prior works define different notions of *accountability* [9], [27], [28] that formalize which agents should be held accountable when a protocol fails to satisfy some properties. These definitions are very general, but have been instantiated for selected voting protocols [9], [28], [29], [30]. The accountability properties satisfied by these protocols do not guarantee the resolution of all disputes considered by our dispute resolution properties. We provide a detailed comparison of accountability and our properties in Section VII.

c) Contributions: Our work provides a new foundation for characterizing, reasoning about, and establishing dispute resolution in voting. First, we systematically reason about what disputes can arise in voting for a generic, practically relevant, class of voting protocols. Our class comprises both remote and poll-site voting protocols that can be electronic or paper-based. We then focus on disputes regarding whether the published recorded ballots correctly represent the ballots cast by the voters. Based on our classification, we formally define dispute resolution properties in a symbolic formalism amenable to automated verification using the Tamarin tool [32], [36]. This enables the analysis of a broad class

[1]Here dishonesty includes all deviations of the authority from the protocol specification, both due to corruption or to errors.

of protocols with respect to dispute resolution. Moreover, we identify an important new property, which we call *timeliness*, requiring that when a voter's ballot is recorded incorrectly he has convincing evidence of this by the election's end. This property ensures the resolution of disputes that could not be resolved unambiguously in prior work.

Second, we demonstrate that timeliness can only be guaranteed under strong assumptions (for example, some messages must not be lost on the network) by systematically analyzing what communication channels and trust assumptions are necessary and sufficient to satisfy this property. The result is a complete characterization of all topologies in our voting protocol class for which timeliness holds for some protocol. Such a characterization can guide the design of new voting protocols where timeliness should hold, e.g., by identifying and thereby eliminating settings where timeliness is impossible. We formally verify the claimed properties using proofs constructed by Tamarin and pen-and-paper proofs.

Finally, to simplify establishing dispute resolution in practice, we introduce a property, called *Uniqueness*, that can be checked by everyone and guarantees that each recorded ballot was cast by a unique voter. We prove for protocols where voters can cast at most one ballot that *Uniqueness* implies guarantees for voters who abstain from voting. This has the practical consequence that in many protocols, the corresponding guarantees can be proven more easily. We then present as a case study a mixnet-based voting protocol with dispute resolution and prove that our introduced properties hold, as well as standard voting properties such as verifiability and receipt-freeness.

Overall, our results can be used as follows. In addition to specifying what messages are exchanged between the different agents, a voting protocol in our class specifies (i) how the election's result is computed, (ii) which verification steps are performed, and (iii) when the authority conducting the election is considered to have behaved dishonestly. For (i), it is required that each protocol specifies a function *Tally*. For (ii), as voters must be able to check that no ballots were wrongly recorded for them, a function *castBy* must map each ballot to the voter that has (presumably) cast it. Only if this is defined can a voter notice when a ballot was recorded for him that he has not cast. Finally, dispute resolution requires that a protocol defines a dispute resolution procedure such that everyone can agree on (iii). For this purpose, a protocol may specify a set of executions *Faulty* where the authority is considered to have behaved dishonestly and which only depends on public information and can therefore be evaluated by everyone.

Given a protocol with a dispute resolution procedure and a communication topology, our topology characterization can be used to quickly conclude if the given topology is insufficient to achieve the timeliness aspect of dispute resolution. When this is the case, one can immediately conclude that not all dispute resolution properties can be satisfied. When this is not the case, our formal definitions can be used to analyze whether all dispute resolution properties indeed hold in the protocol. Thereby, in protocols where voters can cast at most one ballot, the guarantees for voters who abstain can be established directly or by showing *Uniqueness* and inferring them by our results.

d) Organization: We describe our protocol model in Section II and the class of voting protocols for which we define our properties in Section III. In Section IV, we classify disputes and define our dispute resolution properties. We then analyze in Section V the communication topologies where timeliness can be achieved. In Section VI, we show how dispute resolution can be established in practice, introduce *Uniqueness*, and present our case study. Finally, we discuss related work in Section VII and conclude in Section VIII.

II. Protocol model and system setup

As is standard in model-checking, we model the protocol and adversary as a (global) transition system. Concretely, we use a formalism that also serves as the input language to the Tamarin tool [32]. Our model uses abstractions that ease the specification of communication channels with security properties and trust assumptions. These kinds of abstractions are now fairly standard in protocol specifications. We complement existing abstractions [4] (e.g., authentic and secure channels and parties that satisfy different kinds of trust assumptions) with new abstractions that are relevant for dispute resolution (e.g., reliable channels described in Section II-E2). Our protocol model is inspired by the model in [5] used for e-voting. We first introduce some terminology relevant for voting protocols and then our protocol model.

a) Terminology: We distinguish between *votes* and *ballots*. Whereas a vote is a voter's choice in plain text, a ballot contains the vote and possibly additional information. The ballots' exact design depends on the voting protocol, but it usually consists of the vote cryptographically transformed to ensure the vote's authenticity or confidentiality. When a ballot is sent by the voter, we say it is *cast*. We denote by the *(voting) authority* the entity responsible for collecting and tallying all voters' ballots. Usually, both the list of collected ballots, called the *recorded ballots*, and the *votes in the final tally* are published on a *public bulletin board* that can be accessed by voters and auditors to verify the election's result.

A. Notation and term algebra

We write $[x_i]_{i\in\{1,\ldots,n\}}$ to denote a list of n messages of the same kind. Similarly, we write $[f(x_i, y_i)]_{i\in\{1,\ldots,n\}}$ for a list whose elements have the same form, but may have different values. When the index set is clear from context, we omit the indices, e.g., we write $[x]$ and $[f(x, y)]$ for the above lists, and we write $[x]_i$ and $[f(x, y)]_i$ for the ith element in the lists. Also, we write $x := y$ for the assignment of y to x.

Our model is based on a term algebra $\mathcal{T}$ that is generated from the application of functions in a signature Σ to a set of names $\mathcal{N}$ and variables $\mathcal{V}$. We use the standard notation and equational theory, given in [4, Appendix A]. The symbols we use here are $\langle p_1, p_2\rangle$ for pairing two terms p_1 and p_2, $\mathit{fst}(p)$ and $\mathit{snd}(p)$ for the first and second projection of the pair p, $\mathit{pk}(x)$ for the public key (or the verification key) associated with a private key (or signing key) x, and $\{m\}_{sk}$ for a message m signed with the signing key *sk*. The equational theory contains standard equations, for example pairing and projection obey $fst(\langle p_1, p_2\rangle) = p_1$ and $snd(\langle p_1, p_2\rangle) = p_2$. We sometimes omit the brackets $\langle\rangle$ when tupling is clear from the context.

We extend the term algebra from [4] with the following function symbols and equations. We use $ver(s, k)$ for signature verification, where s is a signed message and k the verification key. When the signature in s is verified with the (matching) verification key k, the function returns the underlying signed message m and otherwise it returns a default value $\perp$. This is modeled by the equations $ver(s, k) = m$, if $s = \{m\}_{sk}$ and $k = pk(sk)$, and $ver(s, k) = \perp$ otherwise.

Moreover, we use the function *Tally* to model the tallying process in voting. Given a list of ballots $[b]$, $Tally([b])$ denotes the computation of the votes $[v]$ in the final tally, possibly including pre-processing steps such as filtering out invalid ballots. The exact definition of *Tally* depends on the protocol.

Finally, $castBy(b)$ denotes the voter who is considered to be the sender of a ballot b. As with *Tally*, $castBy(b)$ depends on the protocol, in particular on the ballots' design. For example, in a voting protocol where a ballot b contains a voter's identifier (e.g., a code or pseudonym), $castBy(b)$ maps the ballot b to the voter with the identifier included in b. In contrast, in a voting protocol where ballots contain a signature associated with a voter, $castBy(b)$ maps each ballot to the voter associated with the signature contained in b.

As *Tally* and *castBy* are protocol dependent, each concrete protocol specification must define the equations that they satisfy, i.e., extend the term algebra's equational theory with equations characterizing their properties. Note that the functions may not be publicly computable. For example, if only a voter H knows which identifier or signature belongs to him, then other parties are not able to conclude that $castBy(b) = H$.

B. Protocol specification

A protocol consists of multiple *(role) specifications* that define the behavior of the different communicating roles. We model protocols as transition systems that give rise to a trace semantics. Each role specification defines the role's sent and received messages and *signals* that are recorded, ordered sequentially. A signal is a term with a distinguished top-most function symbol. Signals have no effect on a protocol's execution. They merely label events in executions to facilitate specifying the protocol's security properties. We distinguish *explicit signals* that are defined in the specification and *implicit signals* that are recorded during the protocol execution but are not explicitly included in the specification. We explain how we depict protocols as message sequence charts in Section VI-D2.

Roles may possess terms in their *initial knowledge*, which is denoted by the explicit signal *knows*. We require that in a specified role R, any message sent by R must either occur in R's initial knowledge or be deducible from the messages that R initially knows or received in a previous protocol step. Deducibility is defined by the equational theory introduced above. As is standard, whenever R is specified to receive a term that it already has in its knowledge, an agent who instantiates this role will compare the two terms and proceed with the protocol only when they are equal.

C. Adversary model and communication topology

We depict the system setup as a *topology graph* $G = (V, E)$, where the set of vertices V denotes the roles and the set of (directed) edges $E \subseteq (V \times V)$ describes the available communication channels between roles (see e.g. Figure 1). For two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, we define the standard subgraph relation $G_1 \subseteq_G G_2$ as $V_1 \subseteq V_2 \land E_1 \subseteq E_2$.

By default, we consider a Dolev-Yao adversary [19] who has full control over the network, learns all messages sent over the network, and can construct and send messages herself. Additionally, the adversary can *compromise* participating agents to learn their secrets and control their behavior. In a concrete system model, we limit the adversary by trust and channel assumptions. A *(communication) topology* [4] $T = (V, E, t, c)$ specifies the system setup by a graph $G(T) = (V, E)$, trust assumptions by a function $t : V \mapsto trustType$ mapping vertices to trust types, and channel assumptions by a function $c : E \mapsto chanType$ mapping edges to channel types, which denote a channel with certain properties. The types *trustType* and *chanType* are specified in Section II-E, after the execution model. When c is applied to an edge (A, B), we omit duplicate brackets and write $c(A, B)$ instead of $c((A, B))$.

D. Execution model, signals, properties, and assumptions

During protocol execution, roles are instantiated by agents (i.e., the parties involved in the protocol) and we consider all possible interleavings of agents' runs in parallel with the adversary. A *trace* tr is a finite sequence of multisets of the signals associated with an interleaved execution. We denote by $TR(Pr, T)$ the set of all traces of a protocol *Pr* that is *run in the topology* T, i.e., run in parallel with the adversary defined by the topology T.[2] We write $tr_1 \cdot tr_2$ for the concatenation of two traces tr_1 and tr_2.

As previously explained, a trace may contain implicit signals, which are recorded during execution but omitted from the protocol specification for readability, and explicit signals (containing auxiliary information) that we explicitly add to the protocol specification. Implicitly, when an agent A sends a message m (presumably) to B, the signal $send(A, B, m)$ is recorded in the trace. Similarly, when an agent B receives m (presumably) from A, $rec(A, B, m)$ is recorded. Furthermore, the signal $K(m)$ denotes the adversary's knowledge and is recorded whenever the adversary learns a term m and $hon(A)$ is recorded when an honest agent A instantiates a role.

Furthermore, we use the explicit signal $verifyC(H, b)$ to indicate that an honest agent checks whether the ballot b, which was cast by the honest voter H, is recorded correctly (C stands for *cast* and indicates that H cast a ballot). In protocols where voters can cast multiple ballots, this signal

[2] T may specify channels that are never used by *Pr*. Also, *Pr* may specify channels that are not available in T. In the latter case, the corresponding protocol steps cannot be executed and will not occur in the execution.

can occur multiple times for the same voter. Moreover, the signal $\mathit{verifyA}(H, b_H)$ is recorded when an honest agent checks for an honest voter H who has cast the set of ballots b_H, that no ballots other than those in b_H are recorded for H. When this check is done for H who abstained, then $b_H = \emptyset$ (A stands for the fact that H *abstained* from voting). *verifyC* and *verifyA* may be defined such that they can be computed by a machine but not by a human voter, e.g., when they require cryptographic computations. We thus leave it open whether they are performed by the voter H or by another agent such as a helper device.

The explicit signal $\mathit{knows}(A, x)$ is recorded when an agent A has a term x in its initial knowledge. The explicit signals $\mathit{Vote}(H, v)$ and $\mathit{Ballot}(H, b)$ respectively record an honest voter H's vote v and cast ballot b. The former is recorded when H decides what to vote for and the latter is recorded when H casts his ballot. Finally, the explicit signal $\mathit{BB}(m)$ denotes that a message m is published on the bulletin board. We use subscripts to distinguish the signals recorded when different messages are published on the bulletin board. For example, the signals $\mathit{BB}_{\mathit{rec}}([b])$ and $\mathit{BB}_{\mathit{tal}}([v])$ denote that the recorded ballots $[b]$ and the votes in the final tally $[v]$ are published. We will introduce further signals as we need them.

We next define two kinds of trace properties. The first are classical *security properties*, which are specified as sets of traces. A protocol *Pr* run in the topology T, satisfies a security property $\mathcal{S}_S$, if $\mathit{TR}(\mathit{Pr}, T) \subseteq \mathcal{S}_S$. To reason about functional requirements, we additionally define *functional properties*. For example, the empty protocol satisfies many security properties but it is useless for voting because, even in the absence of the adversary, a voter's ballot is never recorded. We will thus require a functional property stating that a protocol must at least have one execution where a voter's ballot is correctly recorded. We describe a functional property by a set of traces $\mathcal{S}_F$, for example containing all traces where a voter's ballot is recorded. We then define that a protocol *Pr* run in the topology T satisfies the property $\mathcal{S}_F$ if $\mathit{TR}(\mathit{Pr}, T) \cap \mathcal{S}_F \neq \emptyset$. Finally, we define protocol *assumptions* as sets of traces. That is, we define so-called *(trace) restrictions* by giving a set of traces and then only consider the traces in the intersection of this set and $\mathit{TR}(\mathit{Pr}, T)$ (see e.g. Section II-E2).

E. Trust and channel types

1) Trust types: In the topologies, we consider four types of trust on roles that reflect the honesty of the agents that execute the role. A *trusted* role means we assume that the agents who instantiate this role are always *honest* and thus strictly follow their role specification. In contrast, an *untrusted* role can be instantiated by *dishonest* agents (i.e., compromised by the adversary) who behave arbitrarily. Dishonest agents model both corrupt entities and entities that unintentionally deviate from their specification, for example due to software errors. We model dishonest agents by sending all their secrets to the adversary and by modeling all their incoming and outgoing channels as insecure (see the channel types below).

In addition, we consider the types *trustFwd* and *trustRpl*, which assume *partial trust*. The agents who instantiate a role of type *trustFwd* or *trustRpl* do not strictly follow their role specification but, respectively, always correctly forward messages or reply upon receiving correct messages. Such assumptions turn out to often be necessary for the timeliness property that we introduce shortly, as otherwise dishonest agents that are expected to forward or answer certain messages can fail to do so and thereby block other protocol participants. Thus, these trust types enable fine grained distinctions to be made about which assumptions are necessary for certain properties to hold.

In summary, we consider the set of trust types $\mathit{trustType} := \{\mathit{trusted}, \mathit{trustFwd}, \mathit{trustRpl}, \mathit{untrusted}\}$. In the topologies, we denote trusted roles by nodes that are circled twice (see e.g., *BB* in Figure 3a, p. 11) and the partial trust types *trustFwd* and *trustRpl* by two dashed circles (see e.g., *P* in Figure 3a). In our protocol class, there is no role that can be mapped both to type *trustFwd* and to type *trustRpl*; thus the interpretation will always be unambiguous. All remaining roles are untrusted.

2) Channel types: In addition to the trust assumptions, a communication topology states channel assumptions. Channels, which are the edges in the topology graph, denote which parties can communicate with each other. Also, channels define assumptions, for example that limit the adversary by stating who can change or learn the messages sent over a given channel. Following Maurer and Schmid [31], we use the notation $A \circ\!\!\rightarrow\!\!\circ B$, $A \bullet\!\!\rightarrow\!\!\circ B$, $A \circ\!\!\rightarrow\!\!\bullet B$, and $A \bullet\!\!\rightarrow\!\!\bullet B$ to denote a channel from (instances of) role A to role B that is respectively insecure, authentic, confidential, and secure. For a formal semantics for these channels, see [4].

We introduce two additional channel assumptions that are useful for dispute resolution. These assumptions concern whether a channel reliably delivers messages and whether external observers can see the communication on a channel. Usually, it is assumed that the above channels are *unreliable* in that the adversary can drop messages sent. We make such assumptions explicit and also allow for *reliable* variants of channels. On a reliable channel, the adversary cannot drop messages and thus all messages sent are received by the intended recipient. We will see in Section V that such channels are needed to achieve timeliness properties.

For dispute resolution, it is sometimes required that external observers can witness the communication an agent is involved in to later judge whether this agent is telling the truth. For example, it may be required that witnesses can observe when a voter casts his physical ballot by placing it into a voting box. Such communication cannot later be denied, e.g. when others witness that the voter has cast his ballot then the voter cannot later deny this. Whereas it is in reality sufficient if several witnesses, e.g., a subset of all voters, can observe such communication, we model this by channels that specify that *any* honest agent can observe such communication. Similarly, we will also model the fact that sufficiently many parties can decide who is right in a dispute by specifying that *any* party can resolve disputes (see Section IV). Concretely, we model communication that can be observed by others by *undeniable* channels where any honest agent $C \notin \{A, B\}$ learns the communication between A and B. This is in contrast to the default *deniable* channels, where an honest agent $C \notin \{A, B\}$ cannot determine that A and B are communicating with each other.

A *channel type* can be built from any combination of the three channel assumptions introduced above. For example, on an insecure reliable channel, the adversary can learn all messages and write messages herself, but she cannot drop messages sent from A to B. However, for dispute resolution not all combinations are useful. In particular, an undeniable channel provides evidence that a message was sent, but this is only useful together with the guarantee that the message is also received. Hence, we only consider undeniable channels that are also reliable. We thus distinguish the following channel types, named after their most significant property: The *default channels* $\circ\!\xrightarrow{d}\!\circ$, $\bullet\!\xrightarrow{d}\!\circ$, $\circ\!\xrightarrow{d}\!\bullet$, $\bullet\!\xrightarrow{d}\!\bullet$, which are neither reliable nor undeniable, the *reliable channels* $\circ\!\xrightarrow{r}\!\circ$, $\bullet\!\xrightarrow{r}\!\circ$, $\circ\!\xrightarrow{r}\!\bullet$, $\bullet\!\xrightarrow{r}\!\bullet$, which are reliable but not undeniable, and the *undeniable channels* $\circ\!\xrightarrow{u}\!\circ$, $\bullet\!\xrightarrow{u}\!\circ$, $\circ\!\xrightarrow{u}\!\bullet$, $\bullet\!\xrightarrow{u}\!\bullet$, which are both reliable and undeniable.

We model the guarantees for senders and receivers that use a reliable or undeniable channel by stating that each message sent on such a channel is also received. We only require this property when both the sender and the receiver of a message are trusted or partially trusted and formally express it by the following restriction.

$$\begin{aligned}\{\mathit{tr} | \forall A, B, m.\ t(A), t(B) \in \{\mathit{trusted}, \mathit{trustFwd}, \mathit{trustRpl}\} \\ \wedge\, c(A,B) \in \{\circ\!\xrightarrow{r}\!\circ, \bullet\!\xrightarrow{r}\!\circ, \circ\!\xrightarrow{r}\!\bullet, \bullet\!\xrightarrow{r}\!\bullet, \circ\!\xrightarrow{u}\!\circ, \bullet\!\xrightarrow{u}\!\circ, \circ\!\xrightarrow{u}\!\bullet, \bullet\!\xrightarrow{u}\!\bullet\} \\ \wedge\, \mathit{send}(A, B, m) \in \mathit{tr} \implies \mathit{rec}(A, B, m) \in \mathit{tr}\}.\end{aligned}$$

To model the additional guarantee that undeniable channels provide, additional signals are recorded in the trace when agents communicate over such channels. That is, whenever an agent A sends a message m to B over an undeniable channel, in addition to the signals $\mathit{send}(A, B, m)$ and $\mathit{rec}(A, B, m)$, the signal $\mathit{Pub}(A, B, m)$ is recorded. We formalize this by the following restriction.

$$\begin{aligned}\{\mathit{tr} | \forall A, B, m.\ c(A,B) \in \{\circ\!\xrightarrow{u}\!\circ, \bullet\!\xrightarrow{u}\!\circ, \circ\!\xrightarrow{u}\!\bullet, \bullet\!\xrightarrow{u}\!\bullet\} \wedge \\ (\mathit{send}(A, B, m) \in \mathit{tr} \vee \mathit{rec}(A, B, m) \in \mathit{tr}) \\ \implies \mathit{Pub}(A, B, m) \in \mathit{tr}\}.\end{aligned}$$

In the rest of this paper, these two restrictions are always stipulated. That is, whenever we use $\mathit{TR}(\mathit{Pr}, T)$ to refer to all traces of the protocol Pr run in the topology T, we actually mean all traces in the intersection of $\mathit{TR}(\mathit{Pr}, T)$ and the above two sets of trace restrictions.

III. Class of voting protocols

Formal reasoning about dispute resolution in voting requires a language for specifying voting protocols and their properties. We provide such a language by presenting a class of voting protocols for which we subsequently define dispute resolution properties. Our class comprises both remote and poll-site voting protocols that can be electronic or paper-based. However, we require a public bulletin board, which is, most of the time, realized by digital means. We define the class by stating natural restrictions that communication topologies and protocols must satisfy to be in our class. Afterwards, we show that many well-known voting protocols belong to this class.

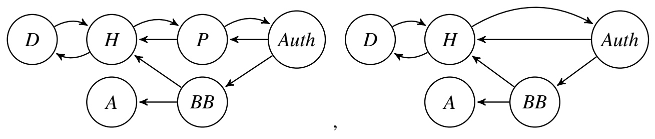


Fig. 1: The topology graphs G_S (left) and G_U (right). We allow for any topology where $G(T) \subseteq_G G_S$ or $G(T) \subseteq_G G_U$.

A. Communication topologies considered

1) Topology graph: The topology graphs in Figure 1 depict all possible roles and communication channels that we consider. That is, we allow for any topology T whose $G(T)$ is a subgraph of G_S or G_U in Figure 1. The node *H* describes two roles for the human voters, one for voters who vote and one for those voters who abstain. Also, there are roles for the devices D and *P*, the voting authority *Auth*, a public bulletin board *BB*, and the auditor *A*. In a concrete protocol, each role, except for *Auth* and *BB*, can be instantiated by multiple agents.

We consider two kinds of setups, G_S and G_U in Figure 1, for two kinds of protocols that differ in how ballots are cast. G_S provides the necessary channels for protocols where each voter H knows his ballot and sends it to *Auth* using a platform *P*. It models remote and poll-site voting. In the former case *P* could be the voter's personal computer, and in the latter case *P* could be a ballot box, or an optical scanner that forwards H's ballot *b* to the authority for tallying. G_U models setups for protocols where a trusted platform P computes (e.g., encrypts) and casts the ballot for H. Often, such protocols do not distinguish between H and P and P operates "in the name of H". Therefore, we model the setup of such protocols by unifying the roles H and P into a single role *voter* H.

In some protocols, voters also have a personal off-line device *D*. In contrast to *P*, *D* has limited capabilities and is not connected to the authority. This models, for example, off-line trusted digital devices or letters containing codes that may be used to compute ballots.

Auth denotes the authority that is responsible for setting up elections and collecting and tallying the ballots. Even though some voting protocols describe the authority in terms of several distinct roles, we collectively describe all these relevant functionalities in a single role, except for the publication of information, which is described by the bulletin board role *BB*. We then also consider just one agent in the role *Auth*. We will argue in Section IV that this is sufficient for our dispute resolution properties. As depicted in Figure 1, *Auth* can publish information on *BB*, which can be read by the auditors and voters. An auditor performs checks on the published information to ensure that the election proceeded correctly. By modeling the auditor as a separate role, this role can be instantiated by anyone, including the voters.

2) Topology assumptions: We further restrict the considered communication topologies by making some minimal channel and trust assumptions. As is common for many voting protocols [5], [15], [16], [23], we model a secure bulletin board and consider its realization as a separate problem. Such a bulletin board can be used to send messages authentically and consistently from the authority to all voters and auditors. We thus assume that the roles *BB* and A exist and are trusted and that the channels from *Auth* to *BB* as well as from *BB* to H and A exist and are default authentic channels. Furthermore, we only use the following partial trust assumptions. *Auth* can be trusted to always reply with a confirmation upon receiving a correct message (type *trustRpl*) and P can be trusted to always forward messages correctly (type *trustFwd*), e.g., a voting machine can be trusted to forward the entered ballots to a remote server. The remaining channel and trust types can respectively be assigned to any channels and roles. Note that we support protocols using anonymous channels (e.g. Civitas [15]) since, for the properties we consider, anonymous channels can be modeled as our default channels.[3]

For dispute resolution, certain guarantees should hold for an honest voter H, even with an untrusted or partially trusted authority and even if all other voters are untrusted. Similarly, the guarantees for the honest authority should hold even when all voters are untrusted. We therefore only consider topologies T where the roles H and *Auth* are untrusted or partially trusted and analyze dispute resolution with respect to three variations of T. We introduce the following notation. We single out a distinguished voter H for whom the security properties are analyzed. Given a topology T, $T^{Auth^+H^+}$ denotes the same topology but where the trust assumptions about *Auth* and the distinguished H are defined by $t(Auth) = trusted$ and $t(H) = trusted$, T^{Auth^+} is as T but with $t(Auth) = trusted$, and T^{H^+} is as T but with $t(H) = trusted$. Note that in all variations, the trust assumptions about the voters other than H are as in T.

B. Voting protocols considered

We next define the voting protocols considered in terms of the protocols' structure and which equations must be specified in the term algebra. Our definition allows for protocols with re-voting, that is, where voters can send several ballots (e.g., [15]). As explained in Section III-A, we allow protocols where the voter H knows and casts his ballot or where a trusted platform P casts the ballot, in which case we unify the roles H and P.

1) Required functions and equations: A protocol specification must define the equation satisfied by $Tally([b])$, defining how the election's result is computed from the list of recorded ballots $[b]$. Similarly, a protocol must define the equation satisfied by $castBy(b)$, which must map each ballot b to a voter, thereby specifying that this voter is considered to have cast the ballot.

2) Protocol's start and end: We assume that the protocol's setup can specify any number of voters and devices and any relation between them, for example that each voter is associated with a unique trusted device. Also, at the protocol's start some public information may be posted on the bulletin board. For example, this might be some election parameters or the list of all eligible voters, denoted by $BB_H([H])$. Furthermore, some agents may know some terms such that these terms (or associated terms) are initially published on the bulletin board or known to other agents. For example, *Auth*'s public key $pk_{Auth} := pk(sk_{Auth})$ can be posted on the bulletin board at the protocol's start whereby *Auth* has the corresponding private key sk_{Auth} in its initial knowledge. We require, however, that at the protocol's start no honest agent knows a voter H's ballot other than the voter himself.

We also assume that an election has two publicly known deadlines that determine the *voting phase's end*, i.e., when ballots can no longer be cast, and the *moment when all relevant information is published*. We denote the latter by the explicit signal *End* in the *BB* role, which is recorded right after the last message relevant for the election is published.

3) Tallying and publication of results: We assume that ballots are collected and tallied by the authority *Auth* and that the protocol allows voters to abstain from voting. Thus, *Auth* starts the tallying process after the voting phase, even if not all voters have cast a ballot. By the election's end, all valid ballots that were received by the authority have been published on the bulletin board together with the votes in the final tally. In protocols with *re-voting*, all ballots are published in the list of recorded ballots and the tallying process is responsible for removing duplicates. Finally, we assume that all messages sent to the bulletin board are immediately published. That is, whenever *BB* receives a message m, the signal $BB(m)$ is recorded in the trace.

C. Examples of protocols in our voting class

Our class comprises well known voting protocols such as Helios [1], Belenios [16], and Civitas [15]. In these protocols, voters can abstain from voting, the bulletin board is assumed to be secure, and the recorded ballots are published on the bulletin board as they were received by the authority. Moreover, even though these protocols all specify different roles and setups, they can each be understood as instantiations of the setups in Figure 1. Belenios and Civitas both have many authority roles, such as registrars and different trustees, which can be understood as our role *Auth*. In Belenios, the *Bulletin Board* also performs some checks and computations. Thus, to cast it in our protocol class, we must additionally interpret those parts of Belenios's *Bulletin Board* as part of our role *Auth* and just the published part of Belenios's *Bulletin Board* as our role *bulletin board BB*.

Note that there are voting protocols, such as BeleniosRF [12], where the recorded ballots are re-encrypted be-

[3]Distinguishing between anonymous and default channels is relevant when analyzing observational equivalence properties such as coercion resistance. However, for our possibility results, we only consider reachability properties.

fore being published on the bulletin board to achieve stronger privacy properties. Such protocols are not in our class.

IV. Dispute Resolution

In voting, the authority conducting the election should behave as expected. That is, if the authority is dishonest, it must be held accountable for this. For elections that are conducted by multiple parties, we require that it is unambiguously detectable when *any* of these parties misbehave, but we do not require that it is detectable *which* of these parties misbehave. This is sufficient to determine when "the system" running the election does not proceed as expected and to take recovery measures when this is the case, such as declaring the election to be invalid. Thus, except for the bulletin board, we model all of the parties involved in conducting an election as one role (and agent) *authority Auth* and require that this agent is held accountable if it does not behave as expected, i.e. does not follow its role specification.

In contrast to the authority, we should not and cannot require that all other parties, notably the voters, behave as expected. In fact, a well-designed voting protocol should still satisfy its expected properties for the honest voters, even when other voters misbehave. We therefore only consider disputes with respect to claims that the voting authority is dishonest.

We first explain why dispute resolution is needed in elections and characterize all relevant disputes. Afterwards, to formalize our dispute resolution properties, we extend our protocol model with additional signals and functions. We then motivate the required properties using our classification and formalize them using the model extensions.

A. Relevant disputes

After an election, all honest participants should agree on the election's outcome. A protocol where any manipulation by the authority can be detected by suitable checks is called *verifiable*. For voting, the gold standard is *end-to-end verifiability* where the final tally consists of the honest voters' votes, tallied correctly and this can all be checked. Often, this property is divided into *universal* and *individual* verifiability. *Universal verifiability* properties denote that some guarantees hold (e.g., the tally is computed correctly) if an auditor performs appropriate checks on bulletin-board data. Any voter or independent third party can serve as an auditor and do such checks. Therefore, if the universal verifiability checks fail, all honest protocol participants will agree on this fact and such checks never give rise to disputes.

Individual verifiability denotes that each voter can verify that his own ballot has been correctly considered in the list of recorded ballots. As only the voter knows which ballot he has cast, this property relies on checks that must (and can only) be done by *each voter himself*. Hence, individual verifiability checks give rise to the following three problems, where a voter claims that the authority is dishonest while other protocol participants cannot determine whether the voter is lying.

(1) A voter is hindered from taking the protocol step where he casts his ballot, in particular he cannot complete one of the preceding protocols steps. There may be technical as well as social reasons for this. For example, the voter may fail to be provided with the necessary credentials in a setup phase or he cannot access a polling station. For generality, we therefore consider disputes regarding the inability to cast a ballot as out of scope of this paper and focus in the following on disputes concerning whether the recorded ballots correctly reflect the ballots cast by the voters.

(2) A voter who successfully cast his ballot is hindered from reaching the verifiability step. For instance, this can happen when recording a ballot requires receiving a confirmation from the authority, which is sent to the voter too late or not at all.

(3) A voter's check whether his ballot is recorded correctly fails. This can happen when a voter detects that one of his cast ballots was not recorded correctly or when he detects that there is a ballot recorded for him that he never sent.

As a result of the above reasoning, based on (3) we distinguish two possible disputes that must be considered in voting protocols, which are depicted in Figure 2. In both disputes, a voter H's and the authority's claim about H's cast ballot differ, where the authority's claim is denoted by the information on the bulletin board. We take the standpoint that the authority is responsible for setting up a working channel to the bulletin board. That is, if messages are not on the bulletin board that should be there, we consider this to be the authority's fault. In the dispute *D1*, a voter H claims that he cast a ballot b, while the authority *Auth* claims that H did not cast b and in the dispute *D2* their claims are reversed. Note that when H claims to have cast b and *Auth* claims that H cast b', this constitutes both a dispute *D1* with respect to the ballot b and a dispute *D2* with respect to b'.

We require that when a voter learns that the authority *Auth* did not record a ballot that he cast, he can convince the other honest participants that *Auth* is dishonest. This is a prerequisite needed to take recovery measures when such manipulations occur. The same must hold when a voter learns that *Auth* recorded a ballot for him that he did not cast. We respectively denote these properties by $\mathit{VoterC}(\mathit{Auth})$ (in dispute *D1*) and $\mathit{VoterA}(\mathit{Auth})$ (in dispute *D2*).

As explained in (2), it is also a problem when a voter who casts a ballot is hindered from reaching his verifiability check in due time. We thus require that a voter who casts a ballot has some timeliness guarantees, namely that by the election's end either his ballot is correctly recorded or he has evidence to convince others that the authority *Auth* is dishonest. We denote this property by $\mathit{TimelyP}(\mathit{Auth})$.

Finally, it is possible that voters lie. Therefore, we require that an honest authority *Auth* is protected from false convictions in any dispute. We denote this by $\mathit{AuthP}(\mathit{Auth})$.

Some protocols support re-voting, where voters can send a set of ballots, all of which are recorded on the bulletin board. In this case, the dispute *D1* denotes that H claims that at least one of his cast ballots is not listed by *Auth*. We thus require that $\mathit{VoterC}(\mathit{Auth})$ and $\mathit{TimelyP}(\mathit{Auth})$ hold for each cast ballot. Dispute *D2* means that H claims not to have cast some of the ballots that are recorded for him. In such a dispute, the

	Voter H claims that H	**Authority** *Auth* claims that H	Properties protecting H	Properties protecting *Auth*
D1	***cast*** *ballot* b	***did not cast*** *ballot* b	$\mathit{VoterC}(\mathit{Auth})$, $\mathit{TimelyP}(\mathit{Auth})$	$\mathit{AuthP}(\mathit{Auth})$
D2	***did not cast*** *ballot* b	***cast*** *ballot* b	$\mathit{VoterA}(\mathit{Auth})$	$\mathit{AuthP}(\mathit{Auth})$

Fig. 2: Possible disputes in voting. The authority's claim is captured by the information on the bulletin board. The respective disputes can be resolved when all properties in the third and fourth columns hold.

voter must be able to convince everyone that too many ballots are recorded for him and that the authority *Auth* is dishonest. This guarantee generalizes the property $\mathit{VoterA}(\mathit{Auth})$, which we will define so that it covers both situations. As before, the disputes *D1* and *D2* can occur simultaneously, for example when H claims he cast the ballots b_1 and b_2 and *Auth* claims that H cast b_2 and b_3.

B. Protocol model for dispute resolution

To formalize dispute resolution for our class of voting protocols, we extend our protocol model from Section II.

It may be required that agents collect evidence to be used in disputes. We use the signal $\mathit{Ev}(b, \mathit{ev})$ to model that the evidence *ev* concerning the ballot b is collected. We model the *forgery of such evidence* by allowing any dishonest agent to claim that any term in its knowledge is evidence. That is, we allow the adversary to perform an action that records $\mathit{Ev}(b, \mathit{ev})$ for any terms b and *ev* such that $K(\langle b, \mathit{ev}\rangle)$.

As we have argued that all honest agents should be able to agree on the outcome of disputes, we do not specify which agents resolve disputes and how the collected evidence must be communicated to them to file disputes. We merely define that a voting protocol can generate evidence such that *any* third party who obtains this evidence can, together with public information, judge whether the authority *Auth* is dishonest. Recall that in poll-site settings, undeniable channels are used to model that *sufficiently many* witnesses can observe the relevant communication in practice. In this context, requiring that any third party can judge whether the authority is dishonest models that sufficiently many parties can decide this in practice. Abstracting away from these details allows us to focus on which evidence and observations are required to resolve disputes, independently of how undeniable channels are realized in practice.

We thus model the verdict of whether *Auth* should be considered dishonest by a publicly verifiable property *Faulty*, which can be specified as part of each voting protocol, independently of any role. $\mathit{Faulty}(\mathit{Auth}, b)$ defines a set of traces where the agent *Auth* is considered to have behaved dishonestly with respect to the ballot b, i.e., b has presumably not been processed according to the protocol in these traces. For example, $\mathit{Faulty}(\mathit{Auth}, b) := \{tr | \exists B, [b].\ \mathit{Pub}(\mathit{Auth}, B, b) \wedge \mathit{BB}_{rec}([b]) \in tr \wedge b \notin [b]\}$ specifies that *Auth* is considered dishonest in all traces where the ballot b was sent from *Auth* to an agent B on an undeniable channel but not included in the recorded ballots $[b]$ published on the bulletin board.

To ensure that the verdict whether a trace is in the set $\mathit{Faulty}(\mathit{Auth}, b)$ is publicly verifiable, the specification of $\mathit{Faulty}(\mathit{Auth}, b)$ must depend just on evidence and public information. We thus state the following requirement.

Requirement 1. $\mathit{Faulty}(\mathit{Auth}, b)$ may only be defined based on the signals *BB*, *Ev*, and *Pub*.

Whereas the above example satisfies this requirement, the set $\{tr | \exists A, B, m, [b].\ \mathit{send}(A, B, m) \wedge \mathit{BB}_{rec}([b]) \in tr \wedge m \notin [b]\}$ is not a valid definition of $\mathit{Faulty}(\mathit{Auth}, b)$, as *send* is not one of the admissible signals.

As a consequence of the above requirement, not all signals in a trace tr are relevant for evaluating whether tr satisfies a given *Faulty* definition. In particular, let $\mathit{pubtr}(tr)$ be a projection that maps a trace tr to the signals in tr whose top-most function symbol is one of *BB*, *Ev*, or *Pub*, while maintaining the order of these signals. Then, it follows from Requirement 1 that for all traces tr_1 and tr_2 such that $\mathit{pubtr}(tr_1) = \mathit{pubtr}(tr_2)$, it holds that $tr_1 \in \mathit{Faulty}(\mathit{Auth}, b)$ iff $tr_2 \in \mathit{Faulty}(\mathit{Auth}, b)$.

C. Formal dispute resolution properties

We now use our extended model to define the dispute resolution properties for our class of voting protocols. We formalize each property from Figure 2 as a set of traces.

First, we consider the property $\mathit{VoterC}(\mathit{Auth})$ that protects an honest voter who detects that one of his cast ballots is not recorded correctly by the authority *Auth*. Intuitively, we require that if this happens, the voter can then convince others that the authority is dishonest. Specifically, the property states that whenever an honest voter H (or one of his devices) reaches the step where he believes that one of his ballots b should be recorded on *BB*, then either this ballot is correctly included in the list of recorded ballots on *BB* or everyone can conclude that the authority *Auth* is dishonest. We define $\mathit{VoterC}(\mathit{Auth})$ as follows (C denotes that a ballot has been cast).

Definition 1.

$$\mathit{VoterC}(\mathit{Auth}) := \{tr \mid \mathit{verifyC}(H, b) \in tr \implies (\exists [b].\ \mathit{BB}_{rec}([b]) \in tr \wedge b \in [b]) \vee tr \in \mathit{Faulty}(\mathit{Auth}, b)\}.$$

Note that, for notational simplicity, here and in the rest of the paper, when using set comprehension notation like $\{x | F(x, \bar{y})\}$, all free variables $\bar{y}$ different from x are universally quantified, i.e., $\{x | \forall \bar{y}.\ F(x, \bar{y})\}$.

The next property, $\mathit{TimelyP}(\mathit{Auth})$, states that an honest voter H who casts a ballot b cannot be prevented from proceeding in the protocol such that his ballot is recorded or, if he is prevented, then he can convince others that the authority *Auth* is dishonest. In particular, a voter's ballot must be recorded or there must exist evidence that the authority is dishonest *within a useful time period*. Note that we do not require that the resolution of disputes must take place before the election's end and there can be a complaint period afterwards. However, we require that the necessary evidence exists by this fixed deadline as otherwise it could be received after the complaint period ended. We now define *TimelyP*(*Auth*).

Definition 2.

$$\begin{aligned}\mathit{TimelyP}(\mathit{Auth}) := \{tr \mid \exists tr', tr''.\ tr = tr' \cdot tr'' \\ \wedge \mathit{Ballot}(H, b) \in tr' \wedge\ \mathit{End} \in tr'' \\ \implies (\exists [b]. \mathit{BB}_{rec}([b]) \in tr' \wedge b \in [b]) \vee tr \in \mathit{Faulty}(\mathit{Auth}, b)\}.\end{aligned}$$

The difference to *VoterC*(*Auth*) (Definition 1) is that we not only require the property when a verifiability check is reached, but whenever all the relevant information is published on the bulletin board (indicated by *End*) and a voter's ballot was cast before this deadline. We illustrate this difference on an example in Section VI-D3.

For abstaining voters we define *VoterA*(*Auth*). It states that when an honest voter H who abstains from voting, or one of H's devices, checks that no ballot is recorded for H, then either this is the case or everyone can be convinced that the authority *Auth* is dishonest. We define this property such that it can also be used in protocols with re-voting, where a voter who cast a set of ballots b_H checks that no additional ballots are wrongly recorded for him. We define *VoterA*(*Auth*) as follows.

Definition 3.

$$\begin{aligned}\mathit{VoterA}(\mathit{Auth}) := \{tr \mid \mathit{verifyA}(H, b_H) \in tr \\ \implies \neg(\exists [b], b.\ \mathit{BB}_{rec}([b]) \in tr \wedge b \in [b] \wedge\ \mathit{castBy}(b) = H \\ \wedge\ b \notin b_H) \vee\ \exists b.\ tr \in \mathit{Faulty}(\mathit{Auth}, b)\}.\end{aligned}$$

Note that *castBy* is just a claim that H has cast a ballot and does not imply that H has actually cast it. For example, in a protocol where ballots contain a voter's identity in plain text and *castBy*(b) is defined to map each ballot to the voter whose identity it contains, everyone can construct a ballot b such that $H = \mathit{castBy}(b)$, even when H has not cast it.

It is possible, of course, that a voter who claims that the authority is dishonest is lying. Thus, for dispute resolution to be fair, it must not only protect the honest voters but also an honest authority. We formalize by *AuthP*(*Auth*) that traces where the authority *Auth* is honest should not be in *Faulty*(*Auth*, b) for any ballot b.

Definition 4.

$$\begin{aligned}\mathit{AuthP}(\mathit{Auth}) := \{tr \mid \mathit{hon}(\mathit{Auth}) \in tr \\ \implies \forall b.\ tr \notin \mathit{Faulty}(\mathit{Auth}, b)\}.\end{aligned}$$

Even though the above properties are stated independently of any adversary model, *VoterC*(*Auth*), *TimelyP*(*Auth*), and *VoterA*(*Auth*) are guarantees for an honest voter H and must hold even when the authority *Auth* and other voters are dishonest. Similarly, *AuthP*(*Auth*) constitutes a guarantee for *Auth* and must hold even if all voters are dishonest. Therefore, we define a *dispute resolution property* by stating what property must be satisfied by a protocol (1) for the honest voter H, i.e., when the protocol is run in a topology where H is honest, and (2) for the honest authority *Auth*. Additionally, it is usually required that a protocol satisfies some functional requirement when the agents are honest. Thus a dispute resolution property also specifies (3) which functional requirement must hold when both *Auth* and the voter H are honest.

Definition 5. Let p_H and p_{Auth} be two security properties that must hold respectively for an honest voter H and the honest authority *Auth* and let p_f be a functional property that must hold when both agents are honest. A protocol *Pr*, executed in a topology T, satisfies the dispute resolution property $\mathit{DR}(\mathit{Pr}, T, p_H, p_{\mathit{Auth}}, p_f)$ iff

$$\begin{aligned}\mathit{TR}(\mathit{Pr}, T^{\mathit{Auth}^+ H^+}) \subseteq p_H \cap p_{\mathit{Auth}} \wedge \mathit{TR}(\mathit{Pr}, T^{H^+}) \subseteq p_H \\ \wedge \mathit{TR}(\mathit{Pr}, T^{\mathit{Auth}^+}) \subseteq p_{\mathit{Auth}} \wedge \mathit{TR}(\mathit{Pr}, T^{\mathit{Auth}^+ H^+}) \cap p_f \neq \emptyset.\end{aligned}$$

For example, $\mathit{DR}(\mathit{Pr}, T, \mathit{VoterC}(\mathit{Auth}) \cap \mathit{TimelyP}(\mathit{Auth}) \cap \mathit{VoterA}(\mathit{Auth}), \mathit{AuthP}(\mathit{Auth}), f)$ denotes that the protocol *Pr* run in the topology T satisfies all previously introduced properties in the required adversary models. That is, it satisfies the properties *VoterC*(*Auth*), *TimelyP*(*Auth*), and *VoterA*(*Auth*) for an honest voter H, the property *AuthP*(*Auth*) for an honest authority *Auth*, and the functional property f for an honest voter and the honest authority (see the next section for an example of a functional property). Another dispute resolution property that we consider in the next Section is $\mathit{DR}(\mathit{Pr}, T, \mathit{TimelyP}(\mathit{Auth}), \mathit{AuthP}(\mathit{Auth}), f)$, which states that the timeliness property *TimelyP*(*Auth*) should hold for an honest voter H while *AuthP*(*Auth*) is preserved for the honest authority *Auth*.

V. Communication topologies and timeliness

For *TimelyP*(*Auth*), it is a problem when messages are lost as some protocol participants may be waiting for these messages and thus cannot proceed in the protocol. Intuitively, timeliness only holds under strong assumptions. We investigate this next by systematically characterizing the assumptions needed for *TimelyP*(*Auth*) to hold in our protocol class.

A. Problem scope

1) Dispute resolution property: We aim at achieving timeliness guarantees for the voters while also maintaining the *AuthP*(*Auth*) property for the authority *Auth*. Furthermore, we are only interested in protocols where a voter's ballot can actually be recorded. To express the third requirement, we formalize a functional property stating that for a given protocol and topology, there must be an execution where an honest voter H casts a ballot b and where this ballot is published in the list of recorded ballots $[b]$ on the bulletin board before the last relevant information is published (indicated by *End*). Moreover, this property must hold when all agents and the network behave honestly. We denote by *honestNetw* the set of traces where all agents follow the protocol and messages are forwarded on all channels unchanged. The required functional property is defined as the following set of traces.

Definition 6.

$$\mathit{Func} := \{tr \mid \exists tr', tr'', H, b, [b].\ tr = tr' \cdot tr'' \wedge \mathit{Ballot}(H,b) \in tr' \wedge \mathit{BB}_{rec}([b]) \in tr' \wedge b \in [b] \wedge \mathit{End} \in tr'' \wedge tr \in \mathit{honestNetw}\}.$$

Given a protocol *Pr* and a topology T, we would like the dispute resolution property $\mathit{DR}(\mathit{Pr}, T, \mathit{TimelyP}(\mathit{Auth}), \mathit{AuthP}(\mathit{Auth}), \mathit{Func})$, which we write for conciseness as $\mathit{TimelyDR}(\mathit{Pr}, T)$.

2) Additional assumptions: In standard protocol models, honest agents can stop executing their role at any time. For timeliness, this might be a problem as other agents may wait for their messages and cannot proceed in the protocol. We thus state the additional assumption that honest agents do not abort the protocol execution prematurely. Similarly, we assume that partially trusted agents execute the required action once they can. Note that agents can still be blocked, e.g., when waiting for messages that are dropped on the network.

Assumption 1. Honest agents always execute all protocol steps possible and agents who instantiate a role that is trusted to forward or answer messages, i.e., partially trusted, perform this respective action once they can.

The assumption implies, for example, that when a role specifies a send event after a receive event, the agent instantiating the role will always perform the second step right after the first one. However, an agent can also be blocked between two consecutive protocol steps, for example when multiple receive events are specified after each other and the agent must wait for all of them.

Under the above assumption, we characterize all topologies from our protocol class for which there exists a protocol that satisfies *TimelyDR*, i.e., the set $\{T \mid \exists \mathit{Pr}.\ \mathit{TimelyDR}(\mathit{Pr}, T)\}$. As others [4], we introduce a partial order on topologies such that a possibility result (i.e., the existence of a protocol such that $\mathit{TimelyDR}(\mathit{Pr}, T)$ holds) for a weaker topology implies a possibility result for a stronger topology. We then characterize the above set by providing the "boundary" topologies, i.e., the minimal topologies satisfying *TimelyDR*.

B. Communication topology hierarchy

We define a partial order on topologies that, given two topologies, orders them with respect to their trust and system assumptions. We first define a partial order on our trust and channel types. For $t, t' \in \mathit{trustType}$, $t' \sqsubseteq t$ denotes that t is a stronger assumption than t'. We thus have that $\mathit{untrusted} \sqsubseteq \mathit{trustFwd} \sqsubseteq \mathit{trusted}$ and $\mathit{untrusted} \sqsubseteq \mathit{trustRpl} \sqsubseteq \mathit{trusted}$. We also define for two channel types c and c' that c makes stronger assumptions than c'. Formally, let $\sqsubseteq_0$ be the relation where $\circ\xrightarrow{x}\circ \sqsubseteq_0 \circ\xrightarrow{x}\bullet \sqsubseteq_0 \bullet\xrightarrow{x}\bullet$ and $\circ\xrightarrow{x}\circ \sqsubseteq_0 \bullet\xrightarrow{x}\circ \sqsubseteq_0 \bullet\xrightarrow{x}\bullet$ for all $x \in \{d, r, u\}$ and $\xrightarrow{d} \sqsubseteq_0 \xrightarrow{r} \sqsubseteq_0 \xrightarrow{u}$ for all $\rightarrow \in \{\circ\!\rightarrow\!\circ, \bullet\!\rightarrow\!\circ, \circ\!\rightarrow\!\bullet, \bullet\!\rightarrow\!\bullet\}$. We overload the symbol $\sqsubseteq$ and, for channel types, we write $\sqsubseteq = \sqsubseteq_0^*$, i.e., $\sqsubseteq$ is the reflexive transitive closure of $\sqsubseteq_0$.

Using the above, for two topologies $T_1 = (V_1, E_1, t_1, c_1)$ and $T_2 = (V_2, E_2, t_2, c_2)$ we say that T_2 makes at least as strong assumptions as T_1 if T_2 uses channel and trust assumptions that are at least as strong as those in T_1 and if T_2's topology graph includes all the roles and communication channels that exist in T_1:

$$T_1 \sqsubseteq T_2 := G(T_1) \subseteq_G G(T_2) \wedge \forall (v_a, v_b) \in E_1.\ c_1(v_a, v_b) \sqsubseteq c_2(v_a, v_b) \wedge \forall v \in V_1.\ t_1(v) \sqsubseteq t_2(v).$$

We show next that defining the relation this way is useful for relating possibility results for different topologies. In particular, if for a topology T it is possible to satisfy $\mathit{TimelyDR}(\mathit{Pr}, T)$ with some protocol, then for all topologies that make stronger assumptions, there is also a protocol that satisfies the property. The lemma is proven in Appendix A1.

Lemma 1. Let $T_I \sqsubseteq T_S$ be topologies in our class.

$$\exists \mathit{Pr}.\ \mathit{TimelyDR}(\mathit{Pr}, T_I) \implies \exists \mathit{Pr}'.\ \mathit{TimelyDR}(\mathit{Pr}', T_S).$$

C. Characterization of topologies enabling TimelyDR

We next present the minimal topologies satisfying *TimelyDR* in our voting protocol class. In combination with the above hierarchy, this allows us to fully characterize all topologies T that enable $\mathit{TimelyDR}(T, \mathit{Pr})$ for some protocol *Pr*.

The minimal topologies are depicted in Figure 3 and denoted by $T_1, \ldots, T_7$. Recall that the agents *BB* and *A* as well as their incoming and outgoing channels have fixed trust assumptions. In all topologies, there are roles for *H*, *P* and *Auth* (respectively for *H* and *Auth* in T_6 and T_7), as this is required to satisfy the functional property (*H* must cast a ballot, *P* must forward it, and *Auth* must publish it on *BB*). All topologies have a reliable path from *H* to *Auth* and additional trust assumptions, such as (partially) trusted roles or undeniable channels. We present some possible real-world interpretations of these topologies in Section VI-A.

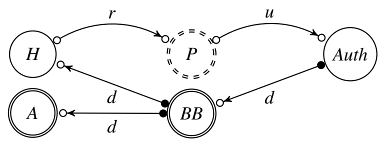
We now state the main theorem for our voting protocol class: The set of topologies for which there exists a protocol that establishes *TimelyDR* consists of all topologies that make at least the assumptions that are made by one of the seven topologies in Figure 3.

Theorem 1. Let $T_1, \ldots, T_7$ be the topologies depicted in Figure 3 and T be a topology in our voting protocol class.

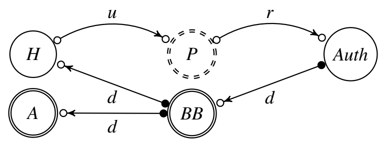
$$(\exists \mathit{Pr}.\ \mathit{TimelyDR}(\mathit{Pr}, T)) \Leftrightarrow (\exists i \in \{1, \ldots, 7\}.\ T_i \sqsubseteq T).$$

We only explain the high level idea of the proof here and refer to Appendix A2 for the details.

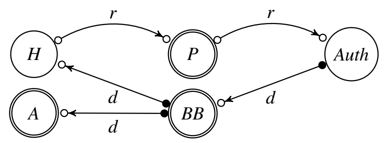
Proof Sketch. First, we establish necessary requirements for topologies to enable *TimelyDR*, by showing by pen-and-paper proofs that any topologies that do not meet these requirements cannot satisfy *TimelyDR* with any protocol. Next, we show that these requirements, which are met by the topologies $T_1, \ldots, T_7$ in Figure 3 are sufficient. In particular, we prove by automated proofs in Tamarin (see [35]) that for each topology $T_i \in \{T_1, \ldots, T_7\}$ there exists a simple protocol Pr_i for which $\mathit{TimelyDR}(T_i, \mathit{Pr}_i)$. Finally, we show (by hand) that the topologies $T_1, \ldots, T_7$ in Figure 3 are the only *minimal* topologies satisfying the necessary and sufficient requirements. It follows that all topologies in our class are either stronger



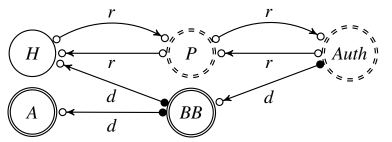
(a) Topology T_1.



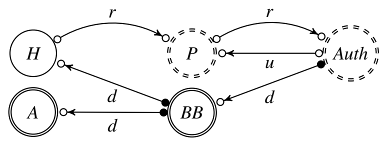
(b) Topology T_2.



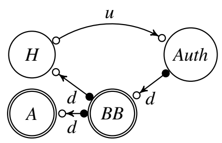
(c) Topology T_3.



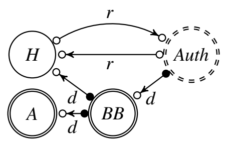
(d) Topology T_4.



(e) Topology T_5.



(f) Topology T_6.



(g) Topology T_7.

Fig. 3: The minimal topologies for which there exists a protocol such that *TimelyDR* can be achieved. The channels' labels denote whether the channels are default (d), reliable (r), or undeniable (u). The nodes' lines denote whether the roles are untrusted (circled once), trusted (circled twice), or partially trusted (dashed circles), where a partially trusted P is of type *trustFwd* and a partially trusted *Auth* is of type *trustRpl*.

than one of the topologies $T_1, \ldots, T_7$ and, by Lemma 1, also also establish *TimelyDR* with some protocol or they are weaker than one of the topologies $T_1, \ldots, T_7$ and thus do not meet the necessary requirements for *TimelyDR*. □

The theorem shows that strong assumptions are indeed necessary to achieve timeliness for dispute resolution. In particular, unreliable channels are insufficient. In most cases, undeniable channels or trusted platforms are required. This can only be avoided in those topologies where there are reliable paths both from the voters to the authority and back. Moreover, *TimelyDR* cannot hold when the platforms are untrusted. This generalizes [2], which states that dispute resolution (called *contestability* in [2]) cannot hold in poll-site voting protocols where ballot-marking devices can be corrupted.

Recall that, in our protocol class, we also allow for offline devices *D*. Our analysis shows that *D* is irrelevant for the question of whether or not *TimelyDR* can be achieved. Also, it is irrelevant whether the channels between the voters and the authority are authentic, confidential, or secure. In particular, they can all be insecure. Nevertheless such devices and channels are needed in voting to satisfy other properties, for example privacy.

VI. Dispute resolution in practice

We now give a practical interpretation of the above results and illustrate how our formalism can be used.

A. Topologies providing TimelyDR

Consider the topologies $T_1, \ldots, T_7$ in Figure 3. In T_1, there is an undeniable channel from *P* to *Auth*. When the platforms are physical ballot boxes, this can be interpreted as the assumption that sufficiently many witnesses see all ballots in the boxes and observe that they are forwarded and considered in the tallying process. The undeniable channel between *H* and *P* in T_2 could, for example, model that witnesses at each polling station observe voters' attempts to cast their ballot, e.g., by scanning their already encrypted ballot on a voting machine [33]. The trusted *P* in T_3 models, for example, that everyone trusts the voting machines used to compute and cast ballots. In this case, the machines can store a trustworthy record of what ballots have been cast for dispute resolution.

In topology T_4, the paths from *H* to *Auth* as well as from *Auth* to *H* are reliable. *P* and *Auth* are respectively trusted to forward and reply. In a remote setting, the reliable channel from H to P could model that voters can always successfully enter messages on their platforms, for example on a working keyboard. The voters could try with several platforms [39] to cast their ballot remotely and receive a confirmation from *Auth* or, in the worst case, go to a physical polling station to do so. The assumptions then model that the voters can find a working platform and website (e.g., public platforms in libraries, polling places, etc.) or they can find a polling station that issues them with a valid confirmation before the election closes. In T_5 and T_6, the undeniable channels could model a distributed ledger on which everyone can respectively observe when confirmations are issued or ballots are cast. Finally, T_7 could model a remote setting similar to T_4, but where ballots are cast by the trusted platforms.

B. Resolving dispute D2 in protocols without re-voting

In practice, the properties *VoterC*$(Auth)$ and *TimelyP*$(Auth)$ can be established in a protocol that provides evidence that a ballot was received by *Auth*. For example, this can be achieved by an undeniable channel or by a confirmation that is sent back from *Auth* to the voter upon a ballot's receipt. *Faulty*$(Auth, b)$ can then be defined as the set of traces where a ballot *b* was received by *Auth* but is not in the set of recorded ballots on the bulletin board (see Section VI-D for a concrete example). In contrast, it is often unclear how *VoterA*$(Auth)$ can be established as a voter who abstains cannot prove the *absence* of a message. To solve this issue, we show that *VoterA*$(Auth)$ is, in many cases, entailed by the *Uniqueness* property, defined next, that can be achieved using standard techniques. We prove this for protocols without re-voting and assume such protocols in the rest of this section.

Uniqueness$(Auth)$ states that whenever any recorded ballots are published and *Auth* is not considered dishonest according to *Faulty*$(Auth, b)$ for some $b \neq \perp$, then each recorded ballot b' has been sent by a unique eligible voter H for which *castBy*$(b') = H$. Thereby, the ballot can be sent as part of a larger composed message. To express that a message m' is a subterm of another message m, we write $m \vdash m'$. As everyone can evaluate *Faulty*$(Auth, b)$, the property's preconditions and thus *Uniqueness*$(Auth)$ are verifiable by everyone.

Definition 7. Let the length of the list $[b]$ be n.

$$\begin{aligned} \mathit{Uniqueness}(\mathit{Auth}) :=\ & \{tr \mid b \neq \perp \wedge tr \notin \mathit{Faulty}(\mathit{Auth}, b) \\ & \wedge BB_{rec}([b]) \in tr \wedge j \in \{1, \ldots, n\} \wedge i \in \{1, \ldots, n\} \implies \\ & \exists [H], i', j', A_1, A_2, m_1, m_2.\ BB_H([H]) \in tr \\ & \wedge \mathit{castBy}([b]_i) = [H]_{i'} \wedge \mathit{castBy}([b]_j) = [H]_{j'} \\ & \wedge \mathit{send}([H]_{i'}, A_1, m_1) \in tr \wedge \mathit{send}([H]_{j'}, A_2, m_2) \in tr \\ & \wedge m_1 \vdash [b]_i \wedge m_2 \vdash [b]_j \wedge (i \neq j \implies [H]_{i'} \neq [H]_{j'})\}. \end{aligned}$$

The property's guarantees are similar to *eligibility verifiability* [26] in that both state that each element of a list on the bulletin board is associated with a unique eligible voter and we compare the two notions in more detail in Appendix C3. Note that the property can only hold for protocols where the list of eligible voters is publicly known.

Intuitively, if a protocol satisfies *Uniqueness*$(Auth)$, then a ballot recorded for the voter H implies that H cast it. Thus, for any ballot that was not cast by H, *Auth* cannot convincingly claim the contrary and an honest voter is thus protected in disputes *D2*. In particular, the traces in the protocol also satisfy *VoterA*$(Auth)$. We prove the following theorem in Appendix B.

Theorem 2. Let *Pr* be a protocol in our class without re-voting and where a voter who abstains does not send any message and let T be a topology in our class.

$$\begin{aligned} & \forall tr \in TR(Pr, T). \\ & tr \in \mathit{Uniqueness}(\mathit{Auth}) \implies tr \in \mathit{VoterA}(\mathit{Auth}). \end{aligned}$$

The theorem has the practical application that, while it is often unclear how *VoterA*$(Auth)$ can be directly realized, *Uniqueness*$(Auth)$ can easily be achieved using standard techniques, such as voters signing their ballots. We provide an example in Section VI-D.

C. How to use our formalism

Given the above results, our formalism can be used to analyze whether a protocol *Pr* and topology T in our class of voting protocols satisfy all dispute resolution properties introduced in Section IV. If there is no topology $T_1, \ldots, T_7$ in Figure 3, such that $T_i \sqsubseteq T$, then we can immediately conclude, by Theorem 1, that *TimelyP*$(Auth)$ and *AuthP*$(Auth)$ cannot hold while the protocol is also functional. Otherwise, analysis is required whether the properties are indeed satisfied by *Pr*.

First, let *Pr* be a protocol that defines a dispute resolution procedure, i.e., specifies the set *Faulty*. Our formalism is mainly intended for this case and can directly be used to analyze whether *VoterC*$(Auth)$, *TimelyP*$(Auth)$, *VoterA*$(Auth)$, and *AuthP*$(Auth)$ hold in such a protocol. In protocols without re-voting that satisfy *Uniqueness*$(Auth)$ and the preconditions of Theorem 2, *VoterA*$(Auth)$ can also be proven by proving *Uniqueness*$(Auth)$ and concluding *VoterA*$(Auth)$ by Theorem 2.

If a protocol *Pr* does not define a dispute resolution procedure *Faulty* then our properties are also undefined. Nevertheless, one can still try to define a verdict *Faulty* using the protocol's specified signals *BB*, *Ev*, and *Pub* and the terms contained in these signals. Our formalism can then be used to establish for each such *Faulty* which properties are satisfied. However, to prove that no definition of *Faulty* achieves dispute resolution, all possible combinations and relations of the above signals and their terms must be considered. Thus, it is in general not straightforward to efficiently conclude that no appropriate definition of *Faulty* exists for a given protocol.

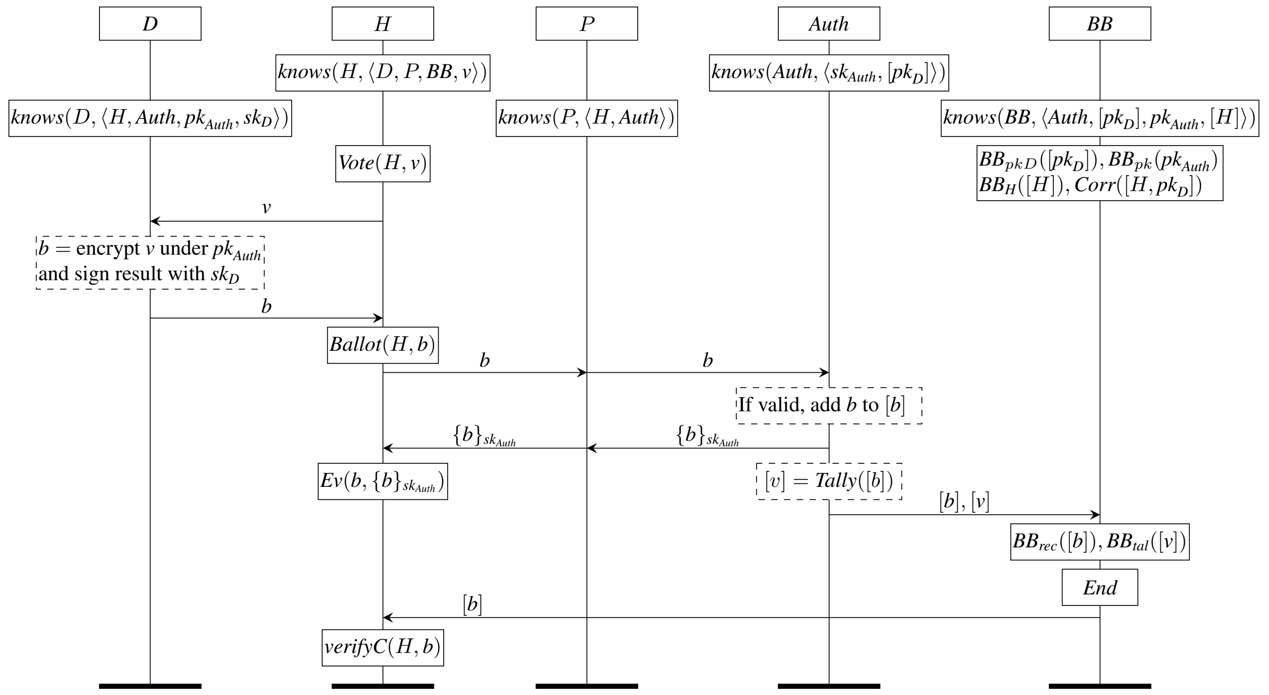
D. A mixnet-based voting protocol with dispute resolution

To demonstrate the applicability of our formalism, we next analyze *MixVote*, a standard mixnet-based voting protocol inspired by [5] with a dispute resolution procedure similar to [30]. In particular, we show how *Faulty* is instantiated, how the properties *VoterC*$(Auth)$ and *TimelyP*$(Auth)$ differ in practice, and that our dispute resolution properties are compatible with standard voting properties, such as verifiability and receipt-freeness. Due to space constraints, we only describe the protocol's main features here, omitting some details such as the auditor's role and the precise definition of some functions and equations in the term algebra. For the detailed protocol specification, the properties' formal definitions, and the proofs we refer to Appendix C

1) Topology: We consider a topology T_{MV} that is as T_4 in Figure 3d, except that there is also a trusted off-line device D, which is connected to the voter H by (bidirectional) secure, default channels. T_{MV} specifies reliable channels between H and the platform P and between P and the authority *Auth*. Also, P and *Auth* are partially trusted to forward messages and reply to messages, respectively. Thus, by Theorem 1, it is possible to achieve *TimelyDR* in the topology T_{MV}.

2) Protocol: We present the protocol as a *message sequence chart*, where each role is depicted by a vertical *life line* and where the box on top names the role. A role's life line denotes the role's events, ordered sequentially. A role's sent and received messages are depicted on top of arrows that start at the sender and end at the recipient. Also, we denote explicit signals by solid squares and the roles' internal computations by dashed squares.

MixVote's simplified specification is depicted in Figure 4. The protocol's setup specifies that at each point in time, only one election takes place (i.e., there are no parallel sessions) and each voter possesses a unique trusted device D to which he has exclusive access. All devices are equipped with a unique signing key sk_D and the authority with a unique secret key sk_{Auth}. The corresponding verification keys from the devices $[pk_D]$ are known to *Auth* and *Auth*'s public key pk_{Auth} is known to all devices. Moreover, all these public keys are published



D
H
P
Auth
BB
knows(H, ⟨D, P, BB, v⟩)
knows(Auth, ⟨sk_Auth, [pk_D]⟩)
knows(D, ⟨H, Auth, pk_Auth, sk_D⟩)
knows(P, ⟨H, Auth⟩)
knows(BB, ⟨Auth, [pk_D], pk_Auth, [H]⟩)
Vote(H, v)
BB_pkD([pk_D]), BB_pk(pk_Auth)
BB_H([H]), Corr([H, pk_D])
v
b = encrypt v under pk_Auth and sign result with sk_D
b
Ballot(H, b)
b
b
If valid, add b to [b]
{b}_sk_Auth
{b}_sk_Auth
Ev(b, {b}_sk_Auth)
[v] = Tally([b])
[b], [v]
BB_rec([b]), BB_tal([v])
End
[b]
verifyC(H, b)

For $b = \perp$: $\mathit{Faulty}(\mathit{Auth}, b) := \{\}$.

For $b \neq \perp$: $\mathit{Faulty}(\mathit{Auth}, b) := \{tr \mid (\exists [b], pk_{Auth}, c.\ BB_{pk}(pk_{Auth}) \in tr \wedge Ev(b,c) \in tr \wedge ver(c, pk_{Auth}) = b \wedge BB_{rec}([b]) \in tr \wedge b \notin [b])$
$\vee (\exists [b], [pk_D].\ BB_{rec}([b]) \in tr \wedge BB_{pkD}([pk_D]) \in tr \wedge$ *not all ballots in* $[b]$ *contain a signature associated with a unique key in* $[pk_D])\}$

Fig. 4: Simplified protocol specification for *MixVote*, without the auditor role and the full function definitions. Here $pk_D = pk(sk_D)$, $pk_{Auth} = pk(sk_{Auth})$, and $castBy(b) = H$ holds iff $\exists pk.\ ver(b, pk) \neq \perp \wedge \langle H, pk \rangle \in [H, pk] \wedge Corr([H, pk]) \in tr$. The protocol's setup specifies a single agent *Auth*, that each voter H is associated with a unique trusted off-line device D, and that there is no restriction on the relation between voters H and platforms P. The role for a voter H who abstains consists of receiving the list of recorded ballots from the bulletin board followed by the signal $\mathit{verifyA}(H, \emptyset)$.

on *BB* (denoted by the signals BB_{pkD} and BB_{pk}, respectively). Additionally, at the protocol's start *BB* knows and publishes the list of eligible voters $[H]$ (denoted by the signal BB_H) and which verification key corresponds to which voter. The latter is denoted by the signal $Corr([H, pk_D])$, where each pair $\langle H, pk_D \rangle$ in the list denotes that the signing key corresponding to pk_D is installed on H's device.

To vote, a voter H uses his device D to compute the ballot as follows: the vote is encrypted under *Auth*'s public key and signed by the device. Then, the voter casts his ballot by entering it on any platform P, which forwards it over the network to *Auth*. For each received ballot b, *Auth* checks b's validity, namely whether b contains a signature corresponding to an eligible voter who has not previously voted. If this is the case, *Auth* adds b to the list of recorded ballots $[b]$. Moreover, as in other protocols [30], to achieve dispute resolution, *Auth* sends back a confirmation to the voter H. The confirmation consists of H's ballot b signed by *Auth* and serves as evidence that b was indeed received by the authority. The voter keeps this confirmation as evidence for later disputes (indicated by the signal *Ev*).

After the voting phase, *Auth* computes the tally from the recorded ballots $[b]$. For this, a standard mixnet is used to decrypt the ballots. This procedure has the properties that no one can learn the correspondence between the encrypted ballots and the decrypted votes. Nevertheless, the mixnet produces evidence, which is published by *Auth* on the bulletin board, that allows everyone to verify that the tally was computed correctly. We describe the detailed functions and equations modeling the *Tally* function in Appendix C1. Also, we describe there the detailed information that is produced by the mixnet and published on *BB* and how an auditor inspects this information to verify the tally.

Among other information, *Auth* publishes on *BB* the recorded ballots $[b]$ and the votes in the final tally $[v]$, as shown in Figure 4. This allows a voter to read the recorded ballots on *BB* and verify that his ballot is included in this list.

A voter who abstains does not send any messages. After the results are published, he reads the list of recorded ballots $[b]$ on *BB* and believes at that step that no ballot should be recorded for him, which is denoted by the signal *verifyA*$(H, \emptyset)$.

We complete the protocol's specification with the definitions of the function *castBy* and the dispute resolution procedure *Faulty*. *castBy* specifies that a ballot b is considered to be cast by the voter H if the ballot's signature can be verified with the verification key that is associated with H. *Faulty* specifies that *Auth* is considered dishonest in all traces where (a) some agent possesses evidence consisting of a ballot b signed by *Auth* that is not included in the recorded ballots $[b]$ on the bulletin board or (b) not all published recorded ballots $[b]$ contain a signature of a unique eligible voter. *castBy* is defined in Figure 4's caption and the description of *Faulty* is given in Figure 4, although we omit here the details of how we model (b).

3) Dispute resolution: Intuitively, by the channel and trust assumptions, each voter who casts a ballot b receives, before the election's end, a confirmation. As this confirmation serves as evidence that b must be on *BB*, *VoterC*(*Auth*) and *TimelyP*(*Auth*) hold. Furthermore, since no one can forge *Auth*'s signature, for a ballot b that was not actually received by *Auth* no one can produce (false) evidence that b should be on *BB*. Thus, *Auth* cannot be falsely convicted and *AuthP*(*Auth*) holds too. Moreover, *Uniqueness*(*Auth*) holds because, when *Faulty* does not hold in an execution, all recorded ballots are signed (and thus were sent) by a unique eligible voter. In particular, *Uniqueness*(*Auth*) implies *VoterA*(*Auth*) in *MixVote*.

To understand the difference between *VoterC*(*Auth*) and *TimelyP*(*Auth*), take a topology T'_{MV} equal to T_{MV} except that *Auth* is untrusted. Assume for simplicity that a voter can interpret whether *Auth*'s signature on the confirmation is valid. In reality, this would require an additional protocol step where the voter uses a device. When the protocol is run in T'_{MV}, it satisfies *VoterC*(*Auth*), as a voter only proceeds with his verifiability check when he has previously received a valid confirmation that convinces everyone that his ballot must be recorded. However, *TimelyP*(*Auth*) is violated as *Auth* may never reply with a valid confirmation and thus block a voter. Consequently, there is an unresolved dispute where an outside observer cannot tell whether a voter did not cast a ballot or the authority did not send a confirmation. In contrast, when the protocol is run in topology T_{MV}, *Auth* always sends a timely response and such disputes do not occur.

4) Standard voting properties: In addition to the dispute resolution properties, we prove in Appendix C that *MixVote* satisfies end-to-end verifiability, consisting of individual verifiability and *tallied-as-recorded*, as well as *eligibility verifiability* [26]. Tallied-as-recorded and eligibility verifiability are two universal verifiability properties that respectively denote that an auditor can verify that the recorded ballots are correctly counted in the final tally and that each vote in the final tally was cast by a unique eligible voter. We also prove that *MixVote* satisfies *receipt-freeness* [18], which denotes that a voter cannot prove to the adversary how he voted, even when he provides the adversary with all secrets that he knows. Intuitively, receipt-freeness holds because the adversary cannot access the voter's device D. Moreover, the evidence used for disputes only contains the ballot and does not reveal the underlying (encrypted) vote.

5) Proofs: We prove in Appendix C4 and by the Tamarin files in [35] that *MixVote* satisfies all above mentioned properties when run in the topology T_{MV}. In particular, we establish most of the properties by automatically proving them for one voter who casts a ballot in Tamarin and by proving them for an arbitrary number of voters by pen-and-paper proofs. The only exceptions are: receipt-freeness, which we prove by Tamarin's built in support for observational equivalence [3]; *VoterA*(*Auth*), which we deduce (by hand) from *Uniqueness*(*Auth*) using Theorem 2; and end-to-end verifiability which we deduce (by hand) from individual verifiability and tallied-as-recorded.

VII. Related work

A. Dispute resolution in poll-site voting protocols

The idea of dispute resolution has been informally considered for *poll-site* voting protocols. In [17], the property considered is called *non-repudiation* and requires that failures *"can not only be detected, but (in most cases) demonstrated"* and that no false convictions can be made. [2] informally considers the properties *contestability* and *defensibility*, which are similar to our dispute resolution properties in that they also protect the honest voters and the honest authority. Contestability requires that some guarantees hold for a voter when he starts the voting process at a polling station. In contrast, our properties *TimelyP*(*Auth*) and *VoterC*(*Auth*) are also suitable for remote settings and respectively make guarantees once a voter casts his ballot and believes that it should be recorded. Moreover, [2]'s definitions are informal and they do not consider timeliness.

In most poll-site voting protocols that consider dispute resolution, voters receive a confirmation as evidence that their ballot was accepted by the authority [7], [10], [11], [13], [17], [24], [38]. In some protocols [10], [17], this confirmation contains the authority's digital signature. In the protocols based on Scantegrity [11], [13], [24], [38] the confirmation consists of a code that is (physically) hidden on the ballot by invisible ink and revealed when a voter marks his choice. A voter's knowledge of a valid code serves as evidence that he voted for a candidate. Thus, when a wrong ballot is recorded, a voter can prove the authority's dishonesty by revealing the code.

Compared to remote voting settings, poll-site protocols profit from the fact that on-site witnesses can observe certain actions. For example, if voters are repeatedly prevented from casting their ballots, this is visible to other voters and auditors in the polling station. Some protocols [24] even explicitly state that voters should publicly declare some decisions before entering them on the voting machine to avoid disputes regarding whether the voting machine correctly followed their instructions. Our notion of undeniable channels allows one to formally consider such assumptions during protocol analysis.

B. Dispute resolution in remote voting protocols

Remotegrity [39] is a remote voting protocol based on Scantegrity, where paper sheets are sent to the voters by postal mail and ballots are cast over the Internet. As with Scantegrity II and III [11], [13], [38], to achieve dispute resolution some codes on these sheets are obscured by a scratch-off surface. If a voter detects a (valid) ballot that is incorrectly recorded for him, he can show to anyone that he has not yet scratched off the relevant codes on his sheets and thus the authority must have falsely recorded this ballot.

[39] discusses several dispute scenarios with respect to whether a ballot is recorded correctly. However, it is stated that *"The [authority] can always force a denial-of-service [..] What Remotegrity does not allow is the [authority] to fully accept (i.e., accept and lock) any ballot the voter did not cast without the voter being able to dispute it."* Thus, the focus is on disputes *D2* in Figure 2, while timeliness in disputes *D1* is not further explored. Moreover, the considered properties as well as the assumed setting are not specified precisely and thus the properties cannot be proven. In contrast, our model enables specifying detailed adversary and system assumptions and provides definitions of dispute resolution properties that can be formally analyzed.

C. Accountability

Our dispute resolution properties are closely related to different notions of *accountability* [9], [27], [28]. Both accountability and our properties formalize how misbehaving protocol participants are identified. While the accountability definitions are generic and allow one to blame different agents in different situations, we focus on understanding what disputes and properties are relevant for voting.

Two accountability definitions have been instantiated for voting protocols. First, accountability due to *Küsters et al.* [28] was instantiated for *Bingo Voting* [8] in [28], for *Helios* [1] in [29], and for *sElect* [30]. These instantiated notions of accountability state that when a defined goal is violated, then some (dishonest) agents can be blamed by a *judge*. A judge may blame multiple parties. As a result, in [29] accountability does not guarantee an unambiguous verdict when a voter claims that his ballot is incorrectly recorded. That is, the property does not guarantee the resolution of such disputes even when the voter is honest. The same holds in [28] and [30] for disputes where a voter claims that he did not receive a required confirmation. To avoid ambiguous verdicts, [28] proposes an alternative accountability property where voters' claims that they did not receive a required confirmation are just ignored. However, this property does not guarantee that the authority is blamed in all situations where an (honest) voter's ballot is not recorded correctly and dispute resolution does not hold.

Second, accountability due to *Bruni et al.* [9] has been instantiated for Bingo Voting in [9]. In this work, *accountability tests* decide whether a given agent should be blamed. However, the accountability test takes as input a ballot and a confirmation that the voter received when casting his ballot. Thus disputes where a voter claims that he cannot receive a confirmation are not considered at all.

In contrast to these two accountability notions, we also consider and resolve disputes where a voter claims that he did not receive a required response from *Auth* after casting the ballot by the property $\mathit{TimelyP}(\mathit{Auth})$. Moreover, our topology characterization allows us to quickly assess when given assumptions are insufficient to satisfy $\mathit{TimelyP}(\mathit{Auth})$.

D. Other related properties

Collection accountability [6] states that when a vote is incorrectly collected, the voter should be provided with evidence to convince an *"independent party"* that this is the case, but it has neither been formally defined nor analyzed. *Dispute freeness* [34] states that there is never a dispute. This property is considered in voting protocols where voters are modeled as machines that conduct an election by engaging in a multiparty protocol [37], [25] and is thus inappropriate for large scale elections where voters must be assumed to have limited computational capabilities. Finally, the FOO protocol [21] allows voters to claim that something went wrong. However, without additional assumptions, FOO does not satisfy our dispute resolution properties. In particular, the signed ballot a voter receives does not prove that the *counter*, who is responsible for tallying, has received the ballot.

VIII. Conclusion

Dispute resolution is an essential ingredient for trustworthy elections and worthy of a careful, formal treatment. Based on a systematic analysis of disputes, we proposed new dispute resolution properties and introduced timeliness as an important aspect thereof. We fully characterized all topologies that achieve timeliness. This provides a formal account for the intuition that timeliness requires strong assumptions. For example, it is not achievable in standard remote voting settings where a network adversary can simply drop messages.

While we have focused on necessary assumptions for dispute resolution, in real elections there are other properties, notably privacy, which may require other assumptions. As future work, we would like to investigate how our topology hierarchy must be adapted for these properties and to characterize the required assumptions for them. The combination of such results with our characterization could lead to new insights about the possibility of achieving different properties simultaneously. Furthermore, such combined results could be a starting point to identify the topologies enabling all properties required in voting; this would help in election design to quickly assess the minimal required setups.

References

[1] Ben Adida. Helios: Web-based Open-audit Voting. In *Proceedings of the 17th Conference on Security Symposium*, SS'08, pages 335–348, Berkeley, CA, USA, 2008. USENIX Association.

[2] Andrew Appel, Richard DeMillo, and Philip Stark. Ballot-Marking Devices (BMDs) Cannot Assure the Will of the Voters. April 21, 2019. Available at SSRN: https://ssrn.com/abstract=3375755 or http://dx.doi.org/10.2139/ssrn.3375755, Accessed: 2019-12-20.

[3] David Basin, Jannik Dreier, and Ralf Sasse. Automated Symbolic Proofs of Observational Equivalence. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 1144–1155. ACM.

[4] David A. Basin, Saša Radomirović, and Michael Schläpfer. A Complete Characterization of Secure Human-Server Communication. In *28th IEEE Computer Security Foundations Symposium, CSF 2015*, pages 199–213. IEEE Computer Society, 2015.

[5] David A. Basin, Saša Radomirović, and Lara Schmid. Alethea: A Provably Secure Random Sample Voting Protocol. In *31th IEEE Computer Security Foundations Symposium, CSF 2018*, pages 283–297. IEEE Computer Society, 2018.

[6] Matthew Bernhard, Josh Benaloh, J. Alex Halderman, Ronald L. Rivest, Peter Y. A. Ryan, Philip B. Stark, Vanessa Teague, Poorvi L. Vora, and Dan S. Wallach. Public Evidence from Secret Ballots. In Robert Krimmer, Melanie Volkamer, Nadja Braun Binder, Norbert Kersting, Olivier Pereira, and Carsten Schürmann, editors, *Electronic Voting*, pages 84–109, Cham, 2017. Springer International Publishing.

[7] Jens-Matthias Bohli, Christian Henrich, Carmen Kempka, Jörn Müller-Quade, and Stefan Röhrich. Enhancing Electronic Voting Machines on the Example of Bingo Voting. *IEEE Trans. Information Forensics and Security*, 4(4):745–750, 2009.

[8] Jens-Matthias Bohli, Jörn Müller-Quade, and Stefan Röhrich. Bingo Voting: Secure and Coercion-Free Voting Using a Trusted Random Number Generator. In Ammar Alkassar and Melanie Volkamer, editors, *E-Voting and Identity*, pages 111–124. Springer Berlin Heidelberg, 2007.

[9] Alessandro Bruni, Rosario Giustolisi, and Carsten Schuermann. Automated Analysis of Accountability. In Phong Q. Nguyen and Jianying Zhou, editors, *Information Security*, pages 417–434. Springer International Publishing, 2017.

[10] Craig Burton, Chris Culnane, James Heather, Thea Peacock, Peter Y. A. Ryan, Steve Schneider, Vanessa Teague, Roland Wen, Zhe Xia, and Sriramkrishnan Srinivasan. Using Prêt à Voter in Victoria State Elections. In *2012 Electronic Voting Technology Workshop / Workshop on Trustworthy Elections, EVT/WOTE '12*, 2012.

[11] Richard Carback, David Chaum, Jeremy Clark, John Conway, Aleksander Essex, Paul S. Herrnson, Travis Mayberry, Stefan Popoveniuc, Ronald L. Rivest, Emily Shen, Alan T. Sherman, and Poorvi L. Vora. Scantegrity II Municipal Election at Takoma Park: The First E2E Binding Governmental Election with Ballot Privacy. In *19th USENIX Security Symposium, Washington, DC, USA, August 11-13, 2010, Proceedings*, USENIX Security'10, pages 291–306. USENIX Association, 2010.

[12] Pyrros Chaidos, Véronique Cortier, Georg Fuchsbauer, and David Galindo. BeleniosRF: A Non-interactive Receipt-Free Electronic Voting Scheme. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 1614–1625. ACM, 2016.

[13] D. Chaum, R. T. Carback, J. Clark, A. Essex, S. Popoveniuc, R. L. Rivest, P. Y. A. Ryan, E. Shen, A. T. Sherman, and P. L. Vora. Scantegrity II: End-to-End Verifiability by Voters of Optical Scan Elections Through Confirmation Codes. *IEEE Transactions on Information Forensics and Security*, 4(4):611–627, 2009.

[14] David Chaum. Random-Sample Voting. http://rsvoting.org/whitepaper/white_paper.pdf, Accessed: 2017-07-07.

[15] Michael R. Clarkson, Stephen Chong, and Andrew C. Myers. Civitas: Toward a Secure Voting System. In *2008 IEEE Symposium on Security and Privacy (S&P 2008)*, pages 354–368, 2008.

[16] Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachène. Election Verifiability for Helios under Weaker Trust Assumptions. In *Computer Security - ESORICS 2014*, pages 327–344. Springer International Publishing, 2014.

[17] Chris Culnane, Peter Y. A. Ryan, Steve Schneider, and Vanessa Teague. vVote: a Verifiable Voting System (DRAFT). *CoRR*, abs/1404.6822, 2014.

[18] Stéphanie Delaune, Steve Kremer, and Mark Ryan. Coercion-Resistance and Receipt-Freeness in Electronic Voting. In *19th IEEE Computer Security Foundations Workshop, (CSFW-19 2006)*, pages 28–42, 2006.

[19] Danny Dolev and Andrew C. Yao. On the Security of Public Key Protocols. *IEEE Trans. Information Theory*, 29(2):198–207, 1983.

[20] Aleksander Essex, Jeremy Clark, Urs Hengartner, and Carlisle Adams. Eperio: Mitigating Technical Complexity in Cryptographic Election Verification. In *Proceedings of the 2010 International Conference on Electronic Voting Technology/Workshop on Trustworthy Elections*, EVT/WOTE'10, pages 1–16. USENIX Association, 2010.

[21] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A Practical Secret Voting Scheme for Large Scale Elections. In *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, ASIACRYPT '92, pages 244–251. Springer-Verlag, 1993.

[22] Jens Groth. A Verifiable Secret Shuffle of Homomorphic Encryptions. *J. Cryptol.*, 23(4):546–579, 2010.

[23] Ari Juels, Dario Catalano, and Markus Jakobsson. Coercion-resistant electronic elections. In David Chaum, Markus Jakobsson, Ronald L. Rivest, Peter A. Ryan, and Josh Benaloh, editors, *Towards Trustworthy Elections, New Directions in Electronic Voting*, pages 37–63. Springer-Verlag, 2010.

[24] Tyler Kaczmarek, John Wittrock, Richard Carback, Alex Florescu, Jan Rubio, Noel Runyan, Poorvi L. Vora, and Filip Zagórski. Dispute Resolution in Accessible Voting Systems: The Design and Use of Audiotegrity. In *E-Voting and Identify*, pages 127–141. Springer Berlin Heidelberg, 2013.

[25] Aggelos Kiayias and Moti Yung. Self-tallying Elections and Perfect Ballot Secrecy. In *Public Key Cryptography*, pages 141–158. Springer Berlin Heidelberg, 2002.

[26] Steve Kremer, Mark Ryan, and Ben Smyth. Election Verifiability in Electronic Voting Protocols. In *Computer Security – ESORICS 2010*, volume 10, pages 389–404. Springer, 2010.

[27] Robert Künnemann, Ilkan Esiyok, and Michael Backes. Automated Verification of Accountability in Security Protocols. In *32nd IEEE Computer Security Foundations Symposium, CSF 2019*, pages 397–413, 2019.

[28] Ralf Küsters, Tomasz Truderung, and Andreas Vogt. Accountability: Definition and Relationship to Verifiability. In *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010*, pages 526–535, 2010.

[29] 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*, pages 395–409, 2012.

[30] R. Küsters, J. Müller, E. Scapin, and T. Truderung. sElect: A Lightweight Verifiable Remote Voting System. In *29th IEEE Computer Security Foundations Symposium CSF 2016*, pages 341–354, 2016.

[31] Ueli M. Maurer and Pierre E. Schmid. A Calculus for Secure Channel Establishment in Open Networks. In *European Symposium on Research in Computer Security*, pages 173–192. Springer, 1994.

[32] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The TAMARIN Prover for the Symbolic Analysis of Security Protocols. In Natasha Sharygina and Helmut Veith, editors, *25th International Conference on Computer Aided Verification (CAV 2013)*, volume 8044 of *LNCS*, pages 696–701. Springer, 2013.

[33] Peter YA Ryan, David Bismark, James A Heather, Steve A Schneider, and Zhe Xia. The Prêt à Voter verifiable election system. *IEEE Transactions on Information Forensics and Security*, 4(4):662–673, 2009.

[34] Krishna Sampigethaya and Radha Poovendran. A framework and taxonomy for comparison of electronic voting schemes. *Computers & Security*, 25(2):137–153, 2006.

[35] Lara Schmid. Tamarin input files. https://github.com/tamarin-prover/tamarin-prover/tree/develop/examples/csf20-disputeResolution.

[36] Benedikt Schmidt, Simon Meier, Cas Cremers, and David Basin. Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties. In *25th IEEE Computer Security Foundations Symposium, CSF 2012*, pages 78–94, 2012.

[37] Berry Schoenmakers. A Simple Publicly Verifiable Secret Sharing Scheme and its Application to Electronic Voting. In *CRYPTO*, pages 148–164. Springer-Verlag, 1999.

[38] Alan T. Sherman, Russell A. Fink, Richard Carback, and David Chaum. Scantegrity III: Automatic Trustworthy Receipts, Highlighting over/Under Votes, and Full Voter Verifiability. In *Proceedings of the 2011 Conference on Electronic Voting Technology/Workshop on Trustworthy Elections*, EVT/WOTE'11, pages 7–7. USENIX Association, 2011.

[39] Filip Zagórski, Richard T. Carback, David Chaum, Jeremy Clark, Aleksander Essex, and Poorvi L. Vora. Remotegrity: Design and Use of an End-to-End Verifiable Remote Voting System. In Michael Jacobson, Michael Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, *Applied Cryptography and Network Security*, pages 441–457. Springer Berlin Heidelberg, 2013.

Appendix

A. Proofs from Section V

We present additional proofs and lemmas for proving the claims from Section V.

1) Topology hierarchy: To prove Lemma 1, we will argue that if two traces or two sets of traces are "similar enough", then they either both satisfy our considered dispute resolution properties or both violate them. To help with this reasoning, we first define a notion of similarity and show two auxiliary lemmas.

Definition 8. Two traces tr_1 and tr_2 are *dispute resolution equal*, denoted by $DRequal(tr_1, tr_2)$, iff for any voter H and ballot b and for the authority $Auth$ it holds that

$$(\exists tr_1', tr_1''.\ tr_1 = tr_1' \cdot tr_1'' \wedge Ballot(H,b) \in tr_1' \wedge End \in tr_1'' \Leftrightarrow \exists tr_2', tr_2''.\ tr_2 = tr_2' \cdot tr_2'' \wedge Ballot(H,b) \in tr_2' \wedge End \in tr_2'') \wedge (hon(Auth) \in tr_1 \Leftrightarrow hon(Auth) \in tr_2) \wedge (pubtr(tr_1) = pubtr(tr_2)).$$

The set of traces $TR(Pr_1, T_1)$ is *dispute resolution similar* to the set of traces $TR(Pr_2, T_2)$, denoted by $DRsimilar(TR(Pr_1, T_1), TR(Pr_2, T_2))$, iff

$$\forall adv \in \{Auth^+H^+, Auth^+, H^+\}, tr_1 \in TR(Pr_1, T_1^{adv}).\ \exists tr_2 \in TR(Pr_2, T_2^{adv}).\ DRequal(tr_1, tr_2).$$

The following auxiliary lemma states that when two traces are dispute resolution equal, then either both satisfy $TimelyP(Auth)$, respectively $AuthP(Auth)$, or both do not satisfy it.

Lemma 2. For two traces tr_1 and tr_2, where $DRequal(tr_1, tr_2)$, it holds that

$$(tr_1 \in TimelyP(Auth) \Leftrightarrow tr_2 \in TimelyP(Auth)) \wedge (tr_1 \in AuthP(Auth) \Leftrightarrow tr_2 \in AuthP(Auth)).$$

Proof. We prove each of the conjuncts separately. Consider two traces tr_1 and tr_2 such that $DRequal(tr_1, tr_2)$. By Definition 2, $TimelyP(Auth)$ holds in a trace tr iff

$$\forall H, b.\ \exists tr', tr''.\ tr = tr' \cdot tr'' \wedge Ballot(H,b) \in tr' \wedge End \in tr'' \implies (\exists [b].\ BB_{rec}([b]) \in tr' \wedge b \in [b]) \vee (tr \in Faulty(Auth, b)).$$

This formula is of the form

$$\forall H, b.\ \exists tr', tr''.\ A(H, b, tr', tr'') \implies B(H, b, tr', tr'') \vee C(H, b).$$

In the following we refer to these predicates simply by A, B, and C. As only the truth values of the predicates A and B depend on the traces tr' and tr'', we consider the formula $(\exists tr', tr''.\ A \implies B) \vee C$, which we will call F, and show that it holds for exactly the same H and b in tr_1 and tr_2. Concretely, we show that if F holds for a given H and b in tr_1, then it also holds for H and b in tr_2. As $DRequal(tr_1, tr_2)$ is symmetric, the same arguments can be applied to show that if F holds for a given H and b in tr_2, then it also holds in tr_1. As this holds for all H and b, it follows that $tr_1 \in TimelyP(Auth)$ iff $tr_2 \in TimelyP(Auth)$.

Let H and b be such that F holds in tr_1. We make a case distinction for the different truth values of A, B, and C.

Case 1): Let H and b be such that $(\exists tr_1', tr_1''.\ A \implies B)$ does not hold in tr_1. As by assumption F holds, it must be the case that C holds, that is $tr_1 \in Faulty(Auth, b)$. By Definition 8, $pubtr(tr_1) = pubtr(tr_2)$ and by Requirement 1 $pubtr(tr_1) = pubtr(tr_2)$ implies that $tr_1 \in Faulty(Auth, b)$ iff $tr_2 \in Faulty(Auth, b)$. Thus, $tr_2 \in Faulty(Auth, b)$ and C also holds in tr_2 for H and b. It follows that F holds in tr_2 for H and b.

Case 2): Let H and b be such that $(\exists tr_1', tr_1''.\ A \implies B)$ holds in tr_1, and A holds and B holds. That is, H and b are such that there exist two traces tr_1' and tr_1'', where

$$tr_1 = tr_1' \cdot tr_1'' \wedge Ballot(H,b) \in tr_1' \wedge End \in tr_1''$$

and such that there exists a list of ballots $[b]$ for which

$$BB_{rec}([b]) \in tr_1' \wedge b \in [b].$$

By Definition 8, there also exist two traces tr_2^1 and tr_2^2, such that

$$tr_2 = tr_2^1 \cdot tr_2^2 \wedge Ballot(H,b) \in tr_2^1 \wedge End \in tr_2^2.$$

Moreover, by Definition 8, $pubtr(tr_1) = pubtr(tr_2)$. As BB_{rec} is a signal in the publicly observable trace and as $BB_{rec}([b])$ with $b \in [b]$ is recorded before End in tr_1, it holds that for two traces tr_2^3 and tr_2^4

$$tr = tr_2^3 \cdot tr_2^4 \wedge BB_{rec}([b]) \in tr_2^3 \wedge b \in [b] \wedge End \in tr_2^4.$$

Thus, as both the signals $Ballot(H,b)$ and $BB_{rec}([b])$ are recorded in tr_2 before End there exist two traces tr_2' and tr_2'' (where tr_2' can be chosen to be the larger trace from tr_2^1 and tr_2^3) such that

$$tr_2 = tr_2' \cdot tr_2'' \wedge Ballot(H,b) \in tr_2' \wedge End \in tr_2'' \wedge BB_{rec}([b]) \in tr_2' \wedge b \in [b].$$

Hence, $(\exists tr_2', tr_2''.\ A \implies B)$ also holds in tr_2 and F is satisfied for H and b.

Case 3): Let H and b be such that $(\exists tr_1', tr_1''.\ A \implies B)$ holds in tr_1 and A does not hold. That is,

$$\exists tr_1', tr_1''.\ \neg(tr_1 = tr_1' \cdot tr_1'' \wedge Ballot(H,b) \in tr_1' \wedge End \in tr_1'').$$

We choose $tr_2' = tr_2'' = \emptyset$ and it holds that

$$\neg(tr_2 = tr_2' \cdot tr_2'' \wedge Ballot(H,b) \in tr_2' \wedge End \in tr_2'').$$

In particular, empty traces cannot contain a signal, thus the second and third conjunct are always false. Thus, it follows that

$$\exists tr_2', tr_2''.\ \neg(tr_2 = tr_2' \cdot tr_2'' \wedge Ballot(H,b) \in tr_2' \wedge End \in tr_2'')$$

and hence A is false in tr_2. It follows that $(\exists tr_2', tr_2''.\ A \implies B)$ and therefore F hold in tr_2 for H and b.

Hence, we showed that in all cases where F holds for a H and b in tr_1, F also holds for H and b in tr_2.

We next prove the lemma's second conjunct. Assume two traces tr_1 and tr_2 such that $\mathit{DRequal}(tr_1, tr_2)$ and assume $tr_1 \in \mathit{AuthP}(\mathit{Auth})$. We show that this implies $tr_2 \in \mathit{AuthP}(\mathit{Auth})$. As *DRequal* is symmetric, the same arguments can be applied to show that $tr_1 \in \mathit{AuthP}(\mathit{Auth})$ follows from $tr_2 \in \mathit{AuthP}(\mathit{Auth})$.

We distinguish two cases for which $tr_1 \in \mathit{AuthP}(\mathit{Auth})$. First, let $\mathit{hon}(\mathit{Auth}) \notin tr_1$. Definition 8 implies $\mathit{hon}(\mathit{Auth}) \notin tr_2$, as $\mathit{hon}(\mathit{Auth}) \in tr_2$ would require $\mathit{hon}(\mathit{Auth}) \in tr_1$, which is a contradiction. From $\mathit{hon}(\mathit{Auth}) \notin tr_2$, it follows that $tr_2 \in \mathit{AuthP}(\mathit{Auth})$ by Definition 4. Second, let $\mathit{hon}(\mathit{Auth}) \in tr_1$ and assume that it holds for all ballots b that $tr_1 \notin \mathit{Faulty}(\mathit{Auth}, b)$. Definition 8 implies $\mathit{hon}(\mathit{Auth}) \in tr_2$. Moreover, by Definition 8 it holds that $\mathit{pubtr}(tr_1) = \mathit{pubtr}(tr_2)$ and, by Requirement 1, $\mathit{pubtr}(tr_1) = \mathit{pubtr}(tr_2)$ implies that $tr_1 \in \mathit{Faulty}(\mathit{Auth}, b)$ iff $tr_2 \in \mathit{Faulty}(\mathit{Auth}, b)$. Thus, it cannot hold that there exists a ballot b for which $tr_2 \in \mathit{Faulty}(\mathit{Auth}, b)$, as this would require $tr_1 \in \mathit{Faulty}(\mathit{Auth}, b)$, which is a contradiction. We thus conclude that for all ballots b, $tr_2 \notin \mathit{Faulty}(\mathit{Auth}, b)$, and thus by Definition 4 $tr_2 \in \mathit{AuthP}(\mathit{Auth})$. □

Using the above lemma, we show the following lemma.

Lemma 3. For two sets of traces $\mathit{TR}(\mathit{Pr}_1, T_1)$ and $\mathit{TR}(\mathit{Pr}_2, T_2)$, where $\mathit{DRsimilar}(\mathit{TR}(\mathit{Pr}_1, T_1), \mathit{TR}(\mathit{Pr}_2, T_2))$, it holds that

$$\begin{aligned}\mathit{TR}(\mathit{Pr}_2, T_2^{\mathit{Auth}^+ H^+}) &\subseteq \mathit{TimelyP}(\mathit{Auth}) \cap \mathit{AuthP}(\mathit{Auth})\\ \wedge \mathit{TR}(\mathit{Pr}_2, T_2^{H^+}) &\subseteq \mathit{TimelyP}(\mathit{Auth})\\ \wedge \mathit{TR}(\mathit{Pr}_2, T_2^{\mathit{Auth}^+}) &\subseteq \mathit{AuthP}(\mathit{Auth}) \implies\\ \mathit{TR}(\mathit{Pr}_1, T_1^{\mathit{Auth}^+ H^+}) &\subseteq \mathit{TimelyP}(\mathit{Auth}) \cap \mathit{AuthP}(\mathit{Auth})\\ \wedge \mathit{TR}(\mathit{Pr}_1, T_1^{H^+}) &\subseteq \mathit{TimelyP}(\mathit{Auth})\\ \wedge \mathit{TR}(\mathit{Pr}_1, T_1^{\mathit{Auth}^+}) &\subseteq \mathit{AuthP}(\mathit{Auth}).\end{aligned}$$

Proof. Assume that

$$\begin{aligned}\mathit{TR}(\mathit{Pr}_2, T_2^{\mathit{Auth}^+ H^+}) &\subseteq \mathit{TimelyP}(\mathit{Auth}) \cap \mathit{AuthP}(\mathit{Auth})\\ \wedge \mathit{TR}(\mathit{Pr}_2, T_2^{H^+}) &\subseteq \mathit{TimelyP}(\mathit{Auth})\\ \wedge \mathit{TR}(\mathit{Pr}_2, T_2^{\mathit{Auth}^+}) &\subseteq \mathit{AuthP}(\mathit{Auth})\end{aligned}$$

and that $\mathit{DRsimilar}(\mathit{TR}(\mathit{Pr}_1, T_1), \mathit{TR}(\mathit{Pr}_2, T_2))$.

Let tr_1 be a trace in $\mathit{TR}(\mathit{Pr}_1, T_1^{\mathit{Auth}^+ H^+})$. By Definition 8, there exists a trace tr_2 in $\mathit{TR}(\mathit{Pr}_2, T_2^{\mathit{Auth}^+ H^+})$ such that $\mathit{DRequal}(tr_1, tr_2)$. By Lemma 2, it holds that $tr_1 \in \mathit{TimelyP}(\mathit{Auth}) \Leftrightarrow tr_2 \in \mathit{TimelyP}(\mathit{Auth})$ and $tr_1 \in \mathit{AuthP}(\mathit{Auth}) \Leftrightarrow tr_2 \in \mathit{AuthP}(\mathit{Auth})$. Since, by assumption, $\mathit{TR}(\mathit{Pr}_2, T_2^{\mathit{Auth}^+ H^+}) \subseteq \mathit{TimelyP}(\mathit{Auth}) \cap \mathit{AuthP}(\mathit{Auth})$ and thus $tr_2 \in \mathit{TimelyP}(\mathit{Auth}) \cap \mathit{AuthP}(\mathit{Auth})$, it follows that $tr_1 \in \mathit{TimelyP}(\mathit{Auth}) \cap \mathit{AuthP}(\mathit{Auth})$. As tr_1 is an arbitrary trace in $\mathit{TR}(\mathit{Pr}_1, T_1^{\mathit{Auth}^+ H^+})$, it follows that $\mathit{TR}(\mathit{Pr}_1, T_1^{\mathit{Auth}^+ H^+}) \subseteq \mathit{TimelyP}(\mathit{Auth}) \cap \mathit{AuthP}(\mathit{Auth})$.

Let tr_1 be a trace in $\mathit{TR}(\mathit{Pr}_1, T_1^{H^+})$. By Definition 8, there exists a trace tr_2 in $\mathit{TR}(\mathit{Pr}_2, T_2^{H^+})$ such that $\mathit{DRequal}(tr_1, tr_2)$. By Lemma 2, it holds that $tr_1 \in \mathit{TimelyP}(\mathit{Auth}) \Leftrightarrow tr_2 \in \mathit{TimelyP}(\mathit{Auth})$. Since, by assumption, $\mathit{TR}(\mathit{Pr}_2, T_2^{H^+}) \subseteq \mathit{TimelyP}(\mathit{Auth})$ and thus $tr_2 \in \mathit{TimelyP}(\mathit{Auth})$, it follows that $tr_1 \in \mathit{TimelyP}(\mathit{Auth})$. As tr_1 is an arbitrary trace in $\mathit{TR}(\mathit{Pr}_1, T_1^{H^+})$, it follows that $\mathit{TR}(\mathit{Pr}_1, T_1^{H^+}) \subseteq \mathit{TimelyP}(\mathit{Auth})$.

Finally, let tr_1 be a trace in $\mathit{TR}(\mathit{Pr}_1, T_1^{\mathit{Auth}^+})$. By Definition 8, there exists a trace tr_2 in $\mathit{TR}(\mathit{Pr}_2, T_2^{\mathit{Auth}^+})$ such that $\mathit{DRequal}(tr_1, tr_2)$. By Lemma 2, it holds that $tr_1 \in \mathit{AuthP}(\mathit{Auth}) \Leftrightarrow tr_2 \in \mathit{AuthP}(\mathit{Auth})$. Since, by assumption, $\mathit{TR}(\mathit{Pr}_2, T_2^{\mathit{Auth}^+}) \subseteq \mathit{AuthP}(\mathit{Auth})$ and thus $tr_2 \in \mathit{AuthP}(\mathit{Auth})$, it follows that $tr_1 \in \mathit{AuthP}(\mathit{Auth})$. As tr_1 is an arbitrary trace in $\mathit{TR}(\mathit{Pr}_1, T_1^{\mathit{Auth}^+})$, it follows that $\mathit{TR}(\mathit{Pr}_1, T_1^{\mathit{Auth}^+}) \subseteq \mathit{AuthP}(\mathit{Auth})$. □

Using the above lemmas, we next prove Lemma 1.

Proof of Lemma 1. We define $T_1 \sqsubset T_2 := T_1 \sqsubseteq T_2 \wedge T_1 \neq T_2$. Let $T_S = (V_S, E_S, t_S, c_S)$, $T_I = (V_I, E_I, t_I, c_I)$, and $T_I \sqsubseteq T_S$ and let *Pr* be an arbitrary protocol such that $\mathit{TimelyDR}(\mathit{Pr}, T_I)$. By Definition 5, $\mathit{TimelyDR}(\mathit{Pr}, T_I) = \mathit{TR}(\mathit{Pr}, T_I^{\mathit{Auth}^+ H^+}) \subseteq \mathit{TimelyP}(\mathit{Auth}) \cap \mathit{AuthP}(\mathit{Auth}) \wedge \mathit{TR}(\mathit{Pr}, T_I^{H^+}) \subseteq \mathit{TimelyP}(\mathit{Auth}) \wedge \mathit{TR}(\mathit{Pr}, T_I^{\mathit{Auth}^+}) \subseteq \mathit{AuthP}(\mathit{Auth}) \wedge \mathit{TR}(\mathit{Pr}, T_I^{\mathit{Auth}^+ H^+}) \cap \mathit{Func} \neq \emptyset$. For two channel or trust types x_I and x_S, we say that x_S is *minimally stronger* than x_I iff $x_I \sqsubset x_S \wedge \neg\exists x_M.\ x_I \sqsubset x_M \sqsubset x_S$. We say that two topologies *differ by one attribute* in the following cases: one topology contains exactly one channel or role that does not exist in the other, a role that occurs in both topologies has a minimally stronger trust type in one of the topologies, or a channel that occurs in both topologies has a minimally stronger channel type in one of the topologies. We distinguish three cases how T_I and T_S, where $T_I \sqsubseteq T_S$, relate: (1) the topologies are equal, (2) they differ only by one attribute, and (3) they differ by several attributes. Case (1), where $T_S = T_I$, is trivial and we look at the other cases.

Case (2): $T_I \sqsubset T_S \wedge \neg\exists T_M.\ T_I \sqsubset T_M \sqsubset T_S$.

T_S can differ from T_I (i) because there is one more (default and insecure) channel or (untrusted) role, (ii) because one of the channel types is stronger, or (iii) because one of the trust types is stronger. We further distinguish between these three cases in (2i)–(2iii).

Case (2i): $(\exists(v_a, v_b).(v_a, v_b) \in E_S \wedge (v_a, v_b) \notin E_I \wedge c_S(v_a, v_b) = \circ\xrightarrow{d}\circ) \vee (\exists v. v \in E_S \wedge v \notin E_I \wedge t_S(v) = \mathit{untrusted})$.

By assumption, $\mathit{TimelyDR}(\mathit{Pr}, T_I)$. Assume that *Pr* only makes use of the vertices and edges in T_I. This is without loss of generality because otherwise we can define a protocol Pr', which is as *Pr* but does not make use of the vertices and edges not contained in T_I and for which $\mathit{TimelyDR}(\mathit{Pr}', T_I)$.

We argue that the traces resulting from *Pr* run in T_S are dispute resolution similar to the traces resulting from *Pr* run in T_I, i.e., $\mathit{DRsimilar}(\mathit{TR}(\mathit{Pr}, T_S), \mathit{TR}(\mathit{Pr}, T_I))$. First, when *Pr* is run in T_S, the honest agents follow the protocol and never send or receive anything on (v_a, v_b). Second, when v_a is under the adversary's control, the adversary can send messages

on (v_a, v_b). However, the adversary sending messages on an insecure channel does not change the fact that the resulting set of traces is dispute resolution similar to the original set $\mathit{TR}(\mathit{Pr}, T_I)$ (it does not change the publicly observable trace, nor the signals *Ballot*, *End*, and *hon*). Moreover, when v_b is honest, it ignores all incoming messages and when v_b is controlled by the adversary too, her knowledge does not differ compared to when the channel (v_a, v_b) does not exist. Thus, the adversary cannot learn, construct, and send messages that were not possible in $\mathit{TR}(\mathit{Pr}, T_I)$. Finally, as the protocol does not specify any behavior for role v, no honest agent will instantiate this role and, as there is no initial knowledge specified for v, the adversary cannot learn any new messages even if she simulates such a role. We conclude that $\mathit{DRsimilar}(\mathit{TR}(\mathit{Pr}, T_S), \mathit{TR}(\mathit{Pr}, T_I))$ and thus, by Lemma 3 and $\mathit{TimelyDR}(\mathit{Pr}, T_I)$, it follows that $\mathit{TR}(\mathit{Pr}, T_S)$ satisfies $\mathit{TimelyP}(\mathit{Auth})$ and $\mathit{AuthP}(\mathit{Auth})$ in the required adversary models.
Moreover, the trace that satisfies *Func* in $\mathit{TR}(\mathit{Pr}, T_I)$ (by assumption such a trace exists) also satisfies *Func* in $\mathit{TR}(\mathit{Pr}, T_S)$, as the same trace is valid even when there is an additional (unused) channel or role.

Case (2ii): $\exists (v_a, v_b) \in E_I . c_I(v_a, v_b) \sqsubseteq c_S(v_a, v_b)$.

We consider the following cases how a channel in T_S is minimally stronger than the same channel in T_I.

(a) $\exists x.\ x \in \{d, r, u\}$
$\wedge\, c_S(v_a, v_b) = \bullet\!\xrightarrow{x}\!\bullet \wedge c_I(v_a, v_b) = \bullet\!\xrightarrow{x}\!\circ$

(b) $\exists x.\ x \in \{d, r, u\}$
$\wedge\, c_S(v_a, v_b) = \bullet\!\xrightarrow{x}\!\bullet \wedge c_I(v_a, v_b) = \circ\!\xrightarrow{x}\!\bullet$

(c) $\exists x.\ x \in \{d, r, u\}$
$\wedge\, c_S(v_a, v_b) = \bullet\!\xrightarrow{x}\!\circ \wedge c_I(v_a, v_b) = \circ\!\xrightarrow{x}\!\circ$

(d) $\exists x.\ x \in \{d, r, u\}$
$\wedge\, c_S(v_a, v_b) = \circ\!\xrightarrow{x}\!\bullet \wedge c_I(v_a, v_b) = \circ\!\xrightarrow{x}\!\circ$

(e) $\exists \rightarrow .\ \rightarrow \in \{\bullet\!\rightarrow\!\bullet, \bullet\!\rightarrow\!\circ, \circ\!\rightarrow\!\bullet, \circ\!\rightarrow\!\circ\}$
$\wedge\, c_S(v_a, v_b) = \xrightarrow{r} \wedge c_I(v_a, v_b) = \xrightarrow{d}$

(f) $\exists \rightarrow .\ \rightarrow \in \{\bullet\!\rightarrow\!\bullet, \bullet\!\rightarrow\!\circ, \circ\!\rightarrow\!\bullet, \circ\!\rightarrow\!\circ\}$
$\wedge\, c_S(v_a, v_b) = \xrightarrow{u} \wedge c_I(v_a, v_b) = \xrightarrow{r}$

We first discuss Cases (a)–(e) and then separately consider the Case (f).

We first argue that for all traces tr and topologies T_I and T_S as described by one of the Cases (a)–(e), it holds that $\mathit{DRsimilar}(\mathit{TR}(\mathit{Pr}, T_S), \mathit{TR}(\mathit{Pr}, T_I))$. This holds as even if the adversary has full control over a channel, she can always behave according to the protocol and not send additional messages, change messages, or reuse messages. Also, she can always deliver messages correctly even on channels that are not reliable. Therefore, the adversary on a weaker channel can perform at least everything that she can on a stronger channel and moreover, the same behavior on a weaker channel does not change any of the signals that are relevant for dispute resolution similarity. We thus conclude by Lemma 3 and by the assumption that $\mathit{TimelyDR}(\mathit{Pr}, T_I)$ that $\mathit{TR}(\mathit{Pr}, T_S)$ satisfies $\mathit{TimelyP}(\mathit{Auth})$ and $\mathit{AuthP}(\mathit{Auth})$ in the required adversary models.

It remains to show, that *Pr* also satisfies the functional property, i.e., $\exists tr.\ tr \in \mathit{TR}(\mathit{Pr}, T_I) \cap \mathit{Func} \implies \exists tr'.\ tr' \in \mathit{TR}(\mathit{Pr}, T_S) \cap \mathit{Func}$. By assumption, the same agents are honest in T_I and T_S. Thus, the same behavior as in tr can be simulated in tr'. Also, on all channels, the adversary only forwards the messages in tr. As $\forall (v_a, v_b).\ (v_a, v_b) \in E_I \implies (v_a, v_b) \in E_S$, the same is possible in T_S, thus the same messages can be sent and the same signals are produced. Therefore, $\mathit{TR}(\mathit{Pr}, T_S) \cap \mathit{Func} \neq \emptyset$ as required.

Finally, we consider the Case (f) separately. T_S and T_I are equal except that (v_a, v_b) is additionally undeniable in T_S. Thus, the same messages can be sent and received with both topologies and the only difference is that if the protocol *Pr* uses the channel (v_a, v_b), then, for some m, $\mathit{Pub}(v_a, v_b, m)$ is additionally recorded in the traces in $\mathit{TR}(\mathit{Pr}, T_S)$ in contrast to the traces in $\mathit{TR}(\mathit{Pr}, T_I)$. Let *Pr* be the protocol such that $\mathit{TimelyDR}(\mathit{Pr}, T_I)$ (which exists by assumption) and Faulty_I be the verdict defined as part of *Pr*. First, consider a ballot b such that the verdict whether a trace tr_P is in $\mathit{Faulty}_I(\mathit{Auth}, b)$ does not depend on whether or not $\mathit{Pub}(v_a, v_b, m) \in tr_P$.

$$
\begin{aligned}
&\forall adv \in \{\mathit{Auth}^+H^+, H^+, \mathit{Auth}^+\}, tr_S \in \mathit{TR}(\mathit{Pr}, T_S^{adv}). \\
&\qquad\qquad \exists tr_I \in \mathit{TR}(\mathit{Pr}, T_I^{adv}). \\
&\qquad tr_I \in \mathit{Faulty}_I(\mathit{Auth}, b) \Leftrightarrow tr_S \in \mathit{Faulty}_I(\mathit{Auth}, b) \\
\overset{(1)}{\Longrightarrow}\, &\forall adv \in \{\mathit{Auth}^+H^+, H^+, \mathit{Auth}^+\}, tr_S \in \mathit{TR}(\mathit{Pr}, T_S^{adv}). \\
&\qquad\qquad \exists tr_I \in \mathit{TR}(\mathit{Pr}, T_I^{adv}). \\
&\qquad (tr_I \in \mathit{TimelyP}(\mathit{Auth}) \Leftrightarrow tr_S \in \mathit{TimelyP}(\mathit{Auth})) \\
&\qquad \wedge (tr_I \in \mathit{AuthP}(\mathit{Auth}) \Leftrightarrow tr_S \in \mathit{AuthP}(\mathit{Auth})) \\
\overset{(2)}{\Longrightarrow}\, &\forall tr_S \in \mathit{TR}(\mathit{Pr}, T_S^{\mathit{Auth}^+H^+}). \\
&\qquad tr_S \in \mathit{TimelyP}(\mathit{Auth}) \cap \mathit{AuthP}(\mathit{Auth}) \\
&\qquad \wedge \forall tr_S \in \mathit{TR}(\mathit{Pr}, T_S^{H^+}).\ tr_S \in \mathit{TimelyP}(\mathit{Auth}) \\
&\qquad \wedge \forall tr_S \in \mathit{TR}(\mathit{Pr}, T_S^{\mathit{Auth}^+}).\ tr_S \in \mathit{AuthP}(\mathit{Auth}).
\end{aligned}
$$

The traces tr_S and tr_I only differ in the signals $\mathit{Pub}(v_a, v_b, m)$ which are irrelevant for $\mathit{Faulty}_I(\mathit{Auth}, b)$ by assumption. Next, the only way the signals $\mathit{Pub}(v_a, v_b, m)$ influence whether a trace is in $\mathit{TimelyP}(\mathit{Auth})$ or $\mathit{AuthP}(\mathit{Auth})$ is if they change the decision whether the trace is in $\mathit{Faulty}_I(\mathit{Auth}, b)$, which is not the case by assumption (Step (1)). Finally, Step (2) holds by the assumption that $\mathit{TimelyDR}(\mathit{Pr}, T_I)$ and thus $\mathit{TR}(\mathit{Pr}, T_I^{\mathit{Auth}^+H^+}) \subseteq \mathit{TimelyP}(\mathit{Auth}) \cap \mathit{AuthP}(\mathit{Auth}) \wedge \mathit{TR}(\mathit{Pr}, T_I^{H^+}) \subseteq \mathit{TimelyP}(\mathit{Auth}) \wedge \mathit{TR}(\mathit{Pr}, T_I^{\mathit{Auth}^+}) \subseteq \mathit{AuthP}(\mathit{Auth})$.

Now consider a ballot b for which the verdict whether tr_P is in $\mathit{Faulty}_I(\mathit{Auth}, b)$ depends on whether or not $\mathit{Pub}(v_a, v_b, m) \in tr_P$. We construct a protocol Pr' that is as *Pr* except that the definition of the verdict for such ballots (that depend on whether or not $\mathit{Pub}(v_a, v_b, m) \in tr_P$) is changed. Namely, $\mathit{Faulty}_S(\mathit{Auth}, b)$ (in Pr') is defined as $\mathit{Faulty}_I(\mathit{Auth}, b)$ except that all occurrences of $\mathit{Pub}(v_a, v_b, m) \in tr_P$ in $\mathit{Faulty}_I(\mathit{Auth}, b)$'s definition are substituted by *false* and all occurrences of $\mathit{Pub}(v_a, v_b, m) \notin tr_P$

by *true*. Then

$$\forall adv \in \{Auth^+H^+, H^+, Auth^+\}, tr_S \in TR(Pr', T_S^{adv}).\ \exists tr_I \in TR(Pr, T_I^{adv}).\ tr_I \in Faulty_I(Auth, b) \Leftrightarrow tr_S \in Faulty_S(Auth, b)$$

as all signals $Pub(v_a, v_b, m)$ that could not influence $Faulty_I(Auth, b)$ (as they did not occur in tr_I), are just ignored by $Faulty_S(Auth, b)$ to simulate the same behavior. We can apply the same Steps as (1) and (2) above, to conclude that Pr' satisfies $TimelyP(Auth)$ and $AuthP(Auth)$ in the required adversary models.

It remains to show that the functional property is preserved. Let $tr \in TR(Pr, T_I) \cap Func$, which exists by assumption. Then, there is a trace tr' in $TR(Pr', T_S)$ where all agents behave exactly as in tr, but which might contain additional signals $Pub(v_a, v_b, m)$ compared to tr. We can conclude that $tr' \in Func$ as *Func* only requires that certain signals appear in the trace and adding more signals does not invalidate the property.

Case (2iii): $\exists v \in V_I . t_I(v) \sqsubseteq t_S(v)$. We consider the following cases:

(a) $t_S(v) = trusted \wedge t_I(v) \in \{trustFwd, trustRpl\}$

(b) $t_S(v) \in \{trustFwd, trustRpl\} \wedge t_I(v) = untrusted$.

Let *Pr* be a protocol such that $TimelyDR(Pr, T_I)$, which exists by assumption. We argue that, in both Cases (a) and (b), $TR(Pr, T_S)$ is dispute resolution similar to $TR(Pr, T_I)$, i.e., $DRsimilar(TR(Pr, T_S), TR(Pr, T_I))$. For this, we take an arbitrary trace tr_S and adversary assumption *adv* such that $tr_S \in TR(Pr, T_S^{adv})$ and argue that there exists a trace tr_I in $TR(Pr, T_I^{adv})$, such that $DRequal(tr_1, tr_2)$.

This is the case as a partially trusted (untrusted) agent in tr_I can always also behave according to the protocol as a trusted (partially trusted) agent in tr_S. Thus the traces tr_S and tr_I denote the same behavior, but may differ in some signals that are only recorded for honest agents, such as *verifyC*. However, all signals *BB* in tr_S are equal to those in tr_I (*BB* is always honest and receives the same messages in tr_I and tr_S), all signals *Ev* are equal in tr_S and tr_I (they can be recorded by untrusted agents), and all signals *Pub* are equal in tr_S and tr_I (as the same channels are used). Thus, $pubtr(tr_I) = pubtr(tr_S)$. Moreover, we never consider topologies where $t_S(H) = trusted$ or $t_S(Auth) = trusted$ for the voter H or the authority *Auth* thus the signals *Ballot* and $hon(Auth)$ are also equal in tr_I and tr_S. We conclude by Definition 8 that $DRequal(tr_1, tr_2)$ and by Lemma 3 that $TR(Pr, T_S)$ satisfies $TimelyP(Auth)$ and $AuthP(Auth)$ with the required adversary models.

Furthermore, it holds that $\exists tr.\ tr \in TR(Pr, T_I) \cap Func \Longrightarrow \exists tr'.\ tr' \in TR(Pr, T_S) \cap Func$, as by definition in *tr* all agents behave according to the protocol which is a valid behavior in T_S that may however result in additional signals being recorded in tr'. However, as additional signals cannot break the property *Func*, tr' satisfies the functional property.

Case (3): $T_I \sqsubset T_S \wedge \exists T_M .\ T_I \sqsubset T_M \sqsubset T_S$.

Let us, step by step and in an arbitrary order, remove one edge, remove one vertex, weaken one channel assumption, and weaken one trust assumption at a time in T_S until we arrive at the topology T_I. As our topologies are finite, i.e., we only consider finitely many roles (vertices), channels (edges), trust types, and channel types, we get a finite sequence of topologies $T_1, \ldots, T_n$ for which $T_I \sqsubseteq T_1 \sqsubseteq \ldots \sqsubseteq T_n \sqsubseteq T_S$ and where each pair of topologies (T_I, T_1), $(T_i, T_{i+1})_{i \in \{1,\ldots,n-1\}}$, and (T_n, T_S) only differs in one attribute. By the results of Case 2, it follows from $T_I \sqsubseteq T_1 \wedge TimelyDR(Pr, T_I)$ that $\exists Pr'.TimelyDR(Pr', T_1)$. We can consecutively apply the result of Case 2 to conclude from $\exists Pr.TimelyDR(Pr, T_i)$ that $\exists Pr'.TimelyDR(Pr', T_{i+1})$, for all $i \in \{1, \ldots, n-1\}$. Finally, we can conclude from $T_n \sqsubseteq T_S \wedge \exists Pr.TimelyDR(Pr, T_n)$ that $\exists Pr'.TimelyDR(Pr', T_S)$, which concludes the proof. ☐

2) Proof of Theorem 1: Next, we state necessary conditions for a topology in our voting protocol class to satisfy *TimelyDR* with some protocol. Then, we show that these conditions are also sufficient. We use this, in combination with the topology hierarchy, to prove Theorem 1 at the end of this section.

Recall from Figure 1 (p. 5) that we distinguish two possible setups and types of protocols: the setup G_S for protocols where the ballots are cast by the voters and the setup G_U, where the voters and the platforms are not distinguished, for protocols that use trusted platforms to cast the ballots. As the topologies modeling these two setups are incomparable, we first consider the necessary conditions for topologies T such that $G(T) \subseteq_G G_S$ and afterwards for the topologies T such that $G(T) \subseteq_G G_U$. As explained in the proof of Lemma 1, we use $T_1 \sqsubset T_2 := T_1 \sqsubseteq T_2 \wedge T_1 \neq T_2$. Also, we sometimes write $T_2 \not\sqsubseteq T_1$ for $\neg(T_2 \sqsubseteq T_1)$.

a) Necessary conditions for T with $G(T) \subseteq_G G_S$: In order to satisfy the functional property, a necessary requirement is the existence of the roles *H*, *P*, and *Auth* and of the channels (H, P) and $(P, Auth)$. This is the case as H must cast a ballot, *P* must forward it, and *Auth* must publish it on the bulletin board.

Given this, we first show that to achieve *TimelyDR* in any topology, there must be a reliable path from *H* to *Auth*. That is, both the channels from *H* to *P* as well as from *P* to *Auth* must be reliable and the platform must be trusted to forward messages correctly. This corresponds to the observation that, in practice, when ballots can be dropped between H and *Auth*, for example when they are cast by mail and not delivered by the post office, then *TimelyDR* cannot hold even when both the voter H and the authority *Auth* are honest.

Lemma 4. Let *Pr* be a protocol and $T = (V, E, t, c)$ be a topology in our voting protocol class such that $G(T) \subseteq_G G_S$, where G_S is the topology graph in Figure 1.

$$TimelyDR(Pr, T) \Longrightarrow \circ\xrightarrow{r}\circ \sqsubseteq c(H, P) \wedge\ \circ\xrightarrow{r}\circ \sqsubseteq c(P, Auth) \wedge\ trustFwd \sqsubseteq t(P).$$

Proof. We assume that there exists a protocol *Pr* that satisfies $TimelyDR(Pr, T)$ for a topology T where (i) $\circ\xrightarrow{r}\circ \not\sqsubseteq c(H, P)$, (ii)$\circ\xrightarrow{r}\circ \not\sqsubseteq c(P, Auth)$, or (iii) $trustFwd \not\sqsubseteq t(P)$ and show that such a protocol cannot exist by arriving at a contradiction. For simplicity, we show the proof for Case (i), but it is analogous for the Cases (ii) and (iii).

By the definition of a dispute resolution property (Definition 5) and by the functional property *Func* (Definition 6), there exists a trace $tr_1 \in TR(Pr, T^{Auth^+ H^+})$ such that $\exists tr_1', tr_1'', H, b, [b].\ tr_1 = tr_1' \cdot tr_1'' \wedge Ballot(H,b) \in tr_1' \wedge BB_{rec}([b]) \in tr_1' \wedge b \in [b] \wedge End \in tr_1'' \wedge tr_1 \in honestNetw$. Let tr_1 be the smallest trace which satisfies this property. As *Auth* is honest in this trace, it further holds that $hon(Auth) \in tr_1$ and since $AuthP(Auth)$ holds, it follows that $\forall b.\ tr_1 \notin Faulty(Auth, b)$.

Suppose that (i) holds. We show that there exists a trace tr_2 in $TR(Pr, T^{Auth^+ H^+})$, which contradicts $DR(Pr, T, TimelyP(Auth), AuthP(Auth), Func)$. In tr_2, the adversary drops all ballots that are sent on the channel between H and P but other than that behaves as in tr_1. This is possible because the channel (H,P) is not reliable in (i). Assume that in tr_2 all honest agents, including H and *Auth*, behave according to the same role as in tr_1 and as specified by *Pr*.

It holds that for some traces tr_2' and tr_2'' and for some list $[b']$, $tr_2 = tr_2' \cdot tr_2'' \wedge Ballot(H,b) \in tr_2' \wedge BB_{rec}([b']) \in tr_2' \wedge b \notin [b'] \wedge End \in tr_2''$. This trace exists since H casts b as in tr_1 and *Auth* publishes only received ballots which are by construction the same as in tr_1 except that *Auth* cannot have received H's ballot b. First, *Auth* cannot receive b from H, as the adversary drops all ballots on the channel between H and P. Moreover, as the adversary does not inject any messages and all honest agents behave as in tr_1, b can also not be received by *Auth* through other channels. Nevertheless, *Auth* follows the same role as in tr_1 and, by the assumptions of our protocol class, publishes the result even if it has not received ballots from all voters. As $b \notin [b']$, for $tr_2 \in TimelyP(Auth)$ it must hold that $tr_2 \in Faulty(Auth, b)$. However, for $tr_2 \in AuthP(Auth)$ it must hold that $tr_2 \notin Faulty(Auth, b)$ as *Auth* is honest. Thus we have a contradiction and such a protocol *Pr*, where $TimelyDR(Pr, T)$ in Case (i), cannot exist.

The other cases can be shown analogously, as the adversary can again drop the ballots that the voter H sends, either on the channel from P to *Auth* (Case (ii)) or on P when it is dishonest (Case (iii)). Note that we allow for protocols with multiple instantiations of the roles. However, by the topology, the channels from and to any instance of P have the same channel type and any instance of P the same trust type. Therefore, the adversary can drop the messages respectively on *all* channels to and from any instance of P or on all instances of P and the above contradiction can be derived independently of the number of instances. □

In addition to the above lemma, a second necessary condition states that a topology must ensure the existence of evidence, which is required in dispute resolution to unequivocally determine whether a ballot under dispute has been received by the authority. This can be achieved under five conditions (the five disjuncts in the next lemma). Evidence can be ensured when one of the channels (H,P) or $(P, Auth)$ is undeniable and generates public evidence that a ballot must have been received by *Auth*. Alternatively, when P is trusted, it can be used to keep a trustworthy record of the cast ballots as evidence. Finally, evidence can be collected in the form of some confirmation that is sent back from *Auth*. This requires *Auth* to be trusted to provide a timely reply and the confirmation must either be sent on an undeniable channel $(Auth, P)$ or it must be ensured that H receives the confirmation by a reliable path from *Auth* to H.

Lemma 5. Let *Pr* be a protocol and $T = (V, E, t, c)$ be a topology in our voting protocol class such that $G(T) \subseteq_G G_S$, where G_S is the topology graph in Figure 1.

$$\begin{aligned} &TimelyDR(Pr, T) \implies \\ &\qquad \circ\xrightarrow{u}\circ \sqsubseteq c(H,P) \vee \circ\xrightarrow{u}\circ \sqsubseteq c(P, Auth) \vee trusted \sqsubseteq t(P) \\ &\qquad \vee (trustRpl \sqsubseteq t(Auth) \wedge \circ\xrightarrow{u}\circ \sqsubseteq c(Auth, P)) \\ &\qquad \vee (trustRpl \sqsubseteq t(Auth) \wedge \circ\xrightarrow{r}\circ \sqsubseteq c(Auth, P) \wedge \circ\xrightarrow{r}\circ \sqsubseteq c(P,H)). \end{aligned}$$

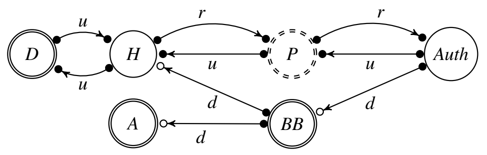
Proof. We show the statement by proving its contrapositive, i.e., we show that if all of the disjuncts are false, then so is the property. We distinguish the three cases where all the disjuncts are false. In all three cases, it holds that the first three disjuncts are false, i.e., $\circ\xrightarrow{u}\circ \not\sqsubseteq c(H,P)$, $\circ\xrightarrow{u}\circ \not\sqsubseteq c(P, Auth)$, and $trusted \not\sqsubseteq t(P)$. Additionally, it holds in Case (i) that $trustRpl \not\sqsubseteq t(Auth)$, in Case (ii) that $\circ\xrightarrow{r}\circ \not\sqsubseteq c(Auth, P)$, and in Case (iii) that $\circ\xrightarrow{u}\circ \not\sqsubseteq c(Auth, P) \wedge \circ\xrightarrow{r}\circ \not\sqsubseteq c(P,H)$. For each case, we take the topology T with the strongest assumptions and where all the conditions of this case hold, i.e. T satisfies the conditions and $\forall T'.\ T'$ *satisfy conditions* $\implies T' \sqsubseteq T$, and show that there cannot exist any protocol *Pr* such that $TimelyDR(Pr, T)$. It follows by Lemma 1 that $\forall T'.\ T'$ *satisfies conditions* $\implies \neg\exists Pr'.\ TimelyDR(Pr', T')$ (Lemma 1 implies that if there were a possibility result for T', there would also be one for T, since $T' \sqsubseteq T$).

Case (i): Figure 5a depicts the maximal topology T_{I1} which satisfies the conditions of this case. We assume that there exists a protocol *Pr* such that $TimelyDR(Pr, T_{I1})$ and show that this leads to a contradiction. By the functional property and the definition of *Func*:

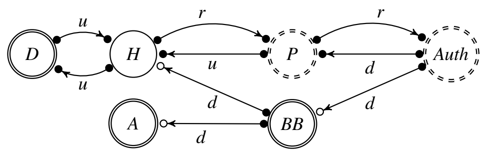
$$\begin{aligned} &\exists tr \in TR(Pr, T_{I1}^{Auth^+ H^+}), tr', tr'', H, b, [b].\ tr = tr' \cdot tr'' \\ &\wedge Ballot(H,b) \in tr' \wedge BB_{rec}([b]) \in tr' \wedge b \in [b] \wedge End \in tr'' \\ &\qquad \wedge tr \in honestNetw. \end{aligned}$$

Let tr be the minimal trace satisfying this. We construct two traces $tr_1 \in TR(Pr, T_{I1}^{H^+})$ and $tr_2 \in TR(Pr, T_{I1}^{Auth^+})$.

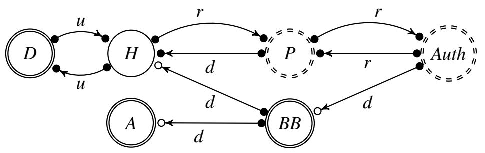
In tr_1, all roles behave exactly as in tr, except for *Auth* who ignores H's ballot b when it is received. In particular, *Auth* does not send any messages on $(Auth, P)$ which depend on the receipt of H's ballot b and does not include b in the published recorded ballots. This is possible as the adversary has full control over *Auth* in this trace and can simulate all send and receive events from tr but leave out selected ones. In tr_2, all roles behave as in tr, except that H, who is dishonest, never casts the ballot b. This is possible as the adversary can



(a) The maximal topology T_{I1} in Case (i).



(b) The maximal topology T_{I2} in Case (ii).



(c) The maximal topology T_{I3} in Case (iii).

Fig. 5: Topologies for which it is impossible to achieve *TimelyDR* by Lemma 5.

simulate all behaviors of *H* in tr up to the point where *H*'s ballot *b* is cast. *Auth* behaves in tr_2 as in tr, however, as it does not receive *H*'s ballot *b*, this is not included in the tallying process. Recall that this complies with any role specification of the authority as, by assumption of our protocol class, the honest *Auth* does not wait for all voters' ballots.

We next reason that the traces can be constructed such that $pubtr(tr_1) = pubtr(tr_2)$, i.e., that all signals whose leading function symbol is one of *Ev*, *Pub*, and *BB* are equal in both traces. First, note that all signals $Ev(b, ev)$ where *ev* only consists of terms initially known by *H*, can be forged by the adversary in tr_2 as she compromises *H* and learns all his secrets. Second, all signals $Ev(b, ev)$ recorded in tr_1 where *ev* does not depend on terms that are only sent by *Auth* if it has received *b*, are also recorded in tr_2. Finally, messages that are sent by *Auth* only when *Auth* has received the ballot *b*, are never sent in tr_1 by assumption and never sent in tr_2 as *Auth* does not receive *b*. Thus, if *ev* depends on such messages, it can neither occur in a recorded signal in tr_1 nor in tr_2.

Similarly, any recorded signal whose leading function symbol is one of *BB* or *Pub* in one trace can be simulated in the other. In particular, all terms t appearing in signals $BB(t)$ or $Pub(Auth, P, t)$ must have been sent by *Auth*. As the dishonest *Auth* in tr_1 has the same knowledge as the honest *Auth* in tr_2, the same terms can be sent to the bulletin board and over the undeniable channel from *Auth* to *P* to generate the same signals. Finally, all signals $Pub(A, B, t)$, for $A \neq Auth$, are equal: either sends on such channels do not depend on *Auth* receiving *H*'s ballot and are done in both traces, or they do depend on it and, in both traces, cannot possibly be done as *Auth* ignores *H*'s ballot in tr_1 and does not receive it in tr_2.

We have shown that $tr_1 \in TR(Pr, T_{I1}^{H^+})$ and $tr_2 \in TR(Pr, T_{I1}^{Auth^+})$ such that $tr = pubtr(tr_1) = pubtr(tr_2)$. By $TimelyP(Auth)$, it must hold that $tr \in Faulty(Auth, b)$, as the honest voter's ballot *b* is not recorded in tr_1. However, by $AuthP(Auth)$, it must hold that $tr \notin Faulty(Auth, b)$ as *Auth* is honest in tr_2, which yields a contradiction. Note that the contradiction can be established independently of the number of devices D and P that can communicate with a voter, as each instantiation of D and P as well as all their incoming and outgoing channels have the same trust and channel types, respectively.

Case (ii): The maximal topology T_{I2} that satisfies these conditions is depicted in Figure 5b. Similarly to Case (i), we assume a protocol *Pr* for which $TimelyDR(Pr, T_{I2})$ and construct two traces $tr_3 \in TR(Pr, T_{I2}^{H^+})$ and $tr_4 \in TR(Pr, T_{I2}^{Auth^+})$ such that $tr = pubtr(tr_3) = pubtr(tr_4)$ and that yield a contradiction as they respectively require that $tr \in Faulty(Auth, b)$ and $tr \notin Faulty(Auth, b)$. In particular, this holds for $tr_4 = tr_2$, where tr_2 is from Case (i), and for tr_3 which is as tr_1 from Case (i) except for the following differences. In tr_3, instead of ignoring H's ballot *b* as in tr_1, the partially trusted *Auth* answers with any response required by the protocol but does not further consider the ballot *b*, e.g., when writing all recorded ballots on the bulletin board. The adversary then drops on the channel $(Auth, P)$ any such responses that are sent by *Auth* only when the ballot *b* from H has been received. This results in $pubtr(tr_3) = pubtr(tr_1)$ and thus the conclusions from Case (i) apply.

Case (iii): The maximal topology T_{I3} that satisfies these conditions is depicted in Figure 5c. Again we assume a protocol *Pr* for which $TimelyDR(Pr, T_{I3})$ and construct two traces $tr_5 \in TR(Pr, T_{I3}^{H^+})$ and $tr_6 \in TR(Pr, T_{I3}^{Auth^+})$ such that $tr = pubtr(tr_5) = pubtr(tr_6)$ and that yield a contradiction as they respectively require that $tr \in Faulty(Auth, b)$ and $tr \notin Faulty(Auth, b)$. This holds for $tr_6 = tr_2$, where tr_2 is from Case (i), and for tr_5 which is as tr_3 from Case (ii), except that the adversary drops all relevant messages sent by *Auth* on the channel (P, H) instead of $(Auth, P)$. The same conclusions as in Cases (i) and (ii) follow. □

b) Necessary conditions for T *with* $G(T) \subseteq_G G_U$*:* Next, we consider the necessary conditions to satisfy *TimelyDR* for topologies where the roles of the voter and platform are unified. We will see that these conditions are closely related to the ones established above. First, note that we require at least the roles H and *Auth* and the channel $(H, Auth)$ as H must cast the ballot and *Auth* must publish it on the bulletin board in order for the functional property to hold.

Next, similarly to Lemma 4, we establish that to achieve *TimelyDR* in any topology, there must be a reliable channel from H to *Auth*. This corresponds to the observation that, in practice, in a remote e-voting setting when ballots are cast over the insecure Internet where they can be dropped, then *TimelyDR* cannot hold even when both the voter H and the authority *Auth* are honest.

Lemma 6. Let *Pr* be a protocol and $T = (V, E, t, c)$ be a topology in our voting protocol class such that $G(T) \subseteq_G G_U$, where G_U is the topology graph in Figure 1.

$$\mathit{TimelyDR}(\mathit{Pr}, T) \implies \circ\xrightarrow{r}\circ \sqsubseteq c(H, \mathit{Auth}).$$

Proof. We assume that there exists a protocol *Pr* and a topology T such that $\mathit{TimelyDR}(\mathit{Pr}, T)$ but where nevertheless $\circ\xrightarrow{r}\circ \not\sqsubseteq c(H, \mathit{Auth})$ and show that such a protocol cannot exist by arriving at a contradiction.
By the definition of a dispute resolution property (Definition 5) and by the functional property *Func* (Definition 6), there exists a trace $tr_1 \in \mathit{TR}(\mathit{Pr}, T^{\mathit{Auth}^+ H^+})$ such that $\exists tr_1', tr_1'', H, b, [b].\ tr_1 = tr_1' \cdot tr_1'' \wedge \mathit{Ballot}(H, b) \in tr_1' \wedge \mathit{BB}_{rec}([b]) \in tr_1' \wedge b \in [b] \wedge \mathit{End} \in tr_1'' \wedge tr_1 \in \mathit{honestNetw}$. Let tr_1 be the smallest trace which satisfies this property.

As in the proof of Lemma 4, we show that there exists a trace tr_2 in $\mathit{TR}(\mathit{Pr}, T^{\mathit{Auth}^+ H^+})$, which contradicts $\mathit{DR}(\mathit{Pr}, T, \mathit{TimelyP}(\mathit{Auth}), \mathit{AuthP}(\mathit{Auth}), \mathit{Func})$. In particular, this holds for a trace tr_2 where all honest agents behave according to the same role as in tr_1 and where the adversary drops all ballots that are sent on the channel between H and *Auth* but other than that behaves as in tr_1.

We can apply the reasoning from the proof of Lemma 4 and conclude that for $tr_2 \in \mathit{TimelyP}(\mathit{Auth})$ it must hold that $tr_2 \in \mathit{Faulty}(\mathit{Auth}, b)$, as the honest voter H's ballot b is not in the list of recorded ballots on the bulletin board in tr_2. However, at the same time, for $tr_2 \in \mathit{AuthP}(\mathit{Auth})$ it must hold that $tr_2 \notin \mathit{Faulty}(\mathit{Auth}, b)$. Thus we have a contradiction and such a protocol *Pr*, where $\mathit{TimelyDR}(\mathit{Pr}, T)$, cannot exist. By the same reasoning as in the proof of Lemma 4, we can also conclude that this holds independently of the number of agents instantiating the roles. □

As for the other setup, we show that a second necessary condition is that a topology ensures the existence of evidence. In particular, evidence can be established by an undeniable channel from H to *Auth*. Alternatively it can be established by a reliable channel (Auth, H) and a partially trusted *Auth*, in which case a confirmation can be sent back from *Auth* to H upon receiving the ballot, which can serve as evidence.

Lemma 7. Let *Pr* be a protocol and $T = (V, E, t, c)$ be a topology in our voting protocol class such that $G(T) \subseteq_G G_U$, where G_U is the topology graph in Figure 1.

$$\mathit{TimelyDR}(\mathit{Pr}, T) \implies \circ\xrightarrow{u}\circ \sqsubseteq c(H, \mathit{Auth})$$
$$\vee(\mathit{trustRpl} \sqsubseteq t(\mathit{Auth}) \wedge \circ\xrightarrow{r}\circ \sqsubseteq c(\mathit{Auth}, H)).$$

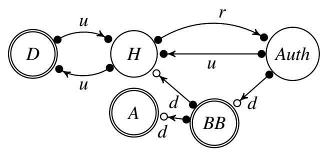
Proof. As in the proof of Lemma 5, we show the statement by proving its contrapositive, i.e., we show that if both disjuncts are false, then so is the property. We distinguish the two cases where the disjuncts are false. In Case (i), $\circ\xrightarrow{u}\circ \not\sqsubseteq c(H, \mathit{Auth})$ and $\mathit{trustRpl} \not\sqsubseteq t(\mathit{Auth})$ and in Case (ii), $\circ\xrightarrow{u}\circ \not\sqsubseteq c(H, \mathit{Auth})$ and $\circ\xrightarrow{r}\circ \not\sqsubseteq c(\mathit{Auth}, H)$. Then, for both cases, we take the topology T with the strongest assumptions and where all the conditions of this case hold and show that there cannot exist any protocol *Pr* such that $\mathit{TimelyDR}(\mathit{Pr}, T)$. As argued in the proof of Lemma 5, this implies an impossibility for all topologies satisfying the conditions (by Lemma 1).

Case (i): The maximal topology T_{I4} that satisfies these conditions is depicted in Figure 6a. As in the proof of Lemma 5, we assume a protocol *Pr* for which $\mathit{TimelyDR}(\mathit{Pr}, T_{I4})$ and construct two traces $tr_1 \in \mathit{TR}(\mathit{Pr}, T_{I4}^{H^+})$ and $tr_2 \in \mathit{TR}(\mathit{Pr}, T_{I4}^{\mathit{Auth}^+})$ such that $tr = \mathit{pubtr}(tr_1) = \mathit{pubtr}(tr_2)$ and that yield a contradiction as they respectively require that $tr \in \mathit{Faulty}(\mathit{Auth}, b)$ and $tr \notin \mathit{Faulty}(\mathit{Auth}, b)$. In particular, let tr_1 be as tr_1 in Lemma 5's proof where *Auth* ignores H's ballot, except that here the fact that *Auth* ignores H's ballot b means that it never sends any message dependent on the receipt of b on the channel (Auth, H) (rather than the channel (Auth, P) in Lemma 5's proof). Also, let tr_2 be as tr_2 in Lemma 5's proof, where H never casts a ballot. As argued in Lemma 5's proof, this results in $\mathit{pubtr}(tr_1) = \mathit{pubtr}(tr_2)$ and yields a contradiction.

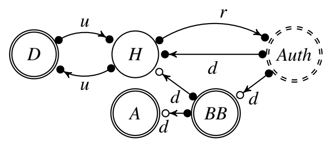
Case (ii): The maximal topology T_{I5} that satisfies these conditions is depicted in Figure 6b. Again we assume a protocol *Pr* for which $\mathit{TimelyDR}(\mathit{Pr}, T_{I5})$ and construct two traces $tr_3 \in \mathit{TR}(\mathit{Pr}, T_{I5}^{H^+})$ and $tr_4 \in \mathit{TR}(\mathit{Pr}, T_{I5}^{\mathit{Auth}^+})$ such that $tr = \mathit{pubtr}(tr_3) = \mathit{pubtr}(tr_4)$ and that yield a contradiction as they respectively require that $tr \in \mathit{Faulty}(\mathit{Auth}, b)$ and $tr \notin \mathit{Faulty}(\mathit{Auth}, b)$. This holds for tr_4 that is as tr_2 from Case (i), and for tr_3 that is as tr_1 from Case (i), except that in tr_3, instead of ignoring H's ballot b, the partially trusted *Auth* answers with any response required by the protocol but does not further consider the ballot b, e.g., when writing all recorded ballots on the bulletin board. The adversary then drops on the channel (Auth, H) any such responses that are sent by *Auth* only when the ballot b from H has been received. This results in $\mathit{pubtr}(tr_3) = \mathit{pubtr}(tr_1)$ and thus the same conclusions as in Case (i) and in the proof of Lemma 5 apply. □

c) Sufficient conditions: We next show that the above conditions are also sufficient by showing that the seven topologies $T_1, \ldots, T_7$ in Figure 3 satisfy these conditions and by establishing a possibility result for each of them. All topologies have a reliable path from H to *Auth*, as this is required by Lemmas 4 and 6, and additional trust assumptions, required by Lemmas 5 and 7.

The next lemma states that for all these topologies, there exists a protocol that satisfies *TimelyDR* and that these topologies are minimal. That is, there are no topologies that have weaker assumptions than $T_1, \ldots, T_7$ but where it is nevertheless possible to achieve *TimelyDR* with some protocol. We establish the lemma's first part by presenting for each topology T_i, $i \in \{1, \ldots, 7\}$ a protocol Pr_i and proving $\mathit{TimelyDR}(\mathit{Pr}_i, T_i)$

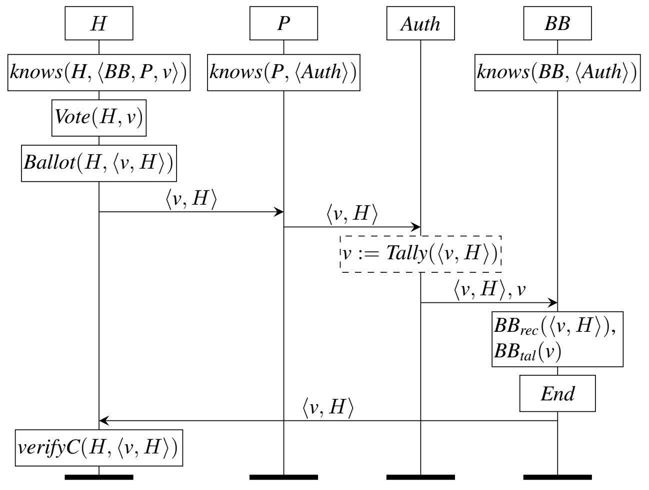


(a) The maximal topology T_{I4} in Case (i).



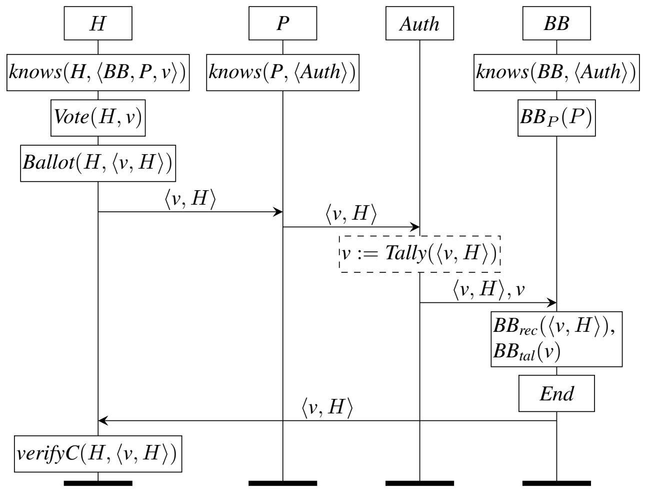
(b) The maximal topology T_{I5} in Case (ii).

Fig. 6: Topologies for which it is impossible to achieve *TimelyDR* by Lemma 7.



$$\mathit{Faulty}(\mathit{Auth}, b) := \{tr \mid \exists P, [b], H, v.\ \mathit{Pub}(P, \mathit{Auth}, b) \in tr \wedge \mathit{BB}_{rec}([b]) \in tr \wedge b \notin [b] \wedge b = \langle v, H\rangle\}.$$

Fig. 7: The protocol Pr_1.



$$\mathit{Faulty}(\mathit{Auth}, b) := \{tr \mid \exists H, P, [b], v.\ \mathit{Pub}(H, P, b) \in tr \wedge \mathit{BB}_P(P) \in tr \wedge \mathit{BB}_{rec}([b]) \in tr \wedge b \notin [b] \wedge b = \langle v, H\rangle\}.$$

Fig. 8: The protocol Pr_2.

with the Tamarin tool [32]. All relevant Tamarin files can be found in [35].

Lemma 8. Let the T_i, for $i \in \{1, \ldots, 7\}$, be the topologies depicted in Figure 3 and T be a topology in our voting protocol class where $T \sqsubset T_i$ for some i.

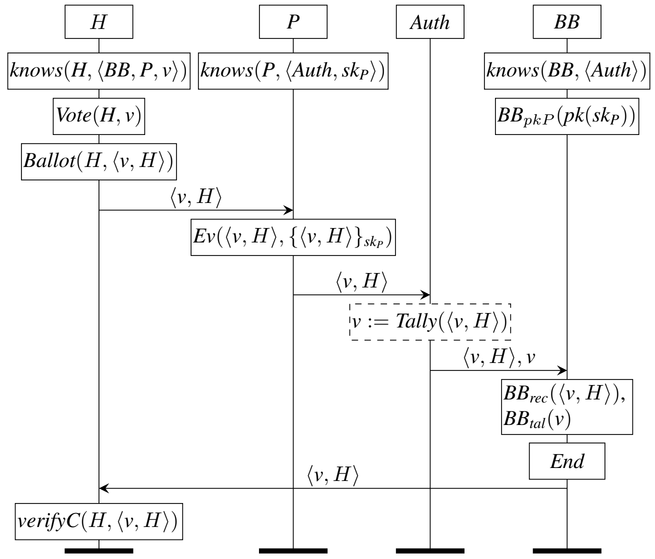
$$\exists \mathit{Pr}.\ \mathit{TimelyDR}(\mathit{Pr}, T_i) \wedge \neg\exists \mathit{Pr}'.\ \mathit{TimelyDR}(\mathit{Pr}', T).$$

Proof. Recall the topologies $T_1, \ldots, T_7$ in Figure 3. We first present for each topology $T_i, i \in \{1, \ldots, 7\}$ a protocol Pr_i, as depicted in Figures 7–13. We present the protocols using message sequence charts, as explained in Section VI-D2. As a simple protocol with one voter is sufficient to demonstrate possibility, all protocols Pr_i specify that only one election is run at a time with one voter H and one platform P. In all protocols, the ballot is a pair consisting of the voter's vote v and his identity H. We extend the term algebra from Section II by defining for the tally function the equation $\mathit{Tally}(\langle v, H\rangle) = v$. Moreover, for each protocol, the definition of *Faulty* is given below the protocol's message sequence chart.

For each topology T_i and protocol Pr_i, we prove that $\mathit{TimelyDR}(\mathit{Pr}_i, T_i)$ holds. To do this, we model in three separate Tamarin theories the traces $\mathit{TR}(\mathit{Pr}_i, T_i^{\mathit{Auth}^+ H^+})$, $\mathit{TR}(\mathit{Pr}_i, T_i^{H^+})$, and $\mathit{TR}(\mathit{Pr}_i, T_i^{\mathit{Auth}^+})$. We then automatically prove that $\mathit{TR}(\mathit{Pr}_i, T_i^{H^+}) \subseteq \mathit{TimelyP}(\mathit{Auth})$, $\mathit{TR}(\mathit{Pr}_i, T_i^{\mathit{Auth}^+}) \subseteq \mathit{AuthP}(\mathit{Auth})$, and $\mathit{TR}(\mathit{Pr}_i, T_i^{\mathit{Auth}^+ H^+}) \subseteq \mathit{TimelyP}(\mathit{Auth}) \cap \mathit{AuthP}(\mathit{Auth})$. All relevant Tamarin files are in [35]. Furthermore, it is obvious that all protocols satisfy $\mathit{TR}(\mathit{Pr}_i, T_i^{\mathit{Auth}^+ H^+}) \cap \mathit{Func} \neq \emptyset$.

Next, we examine for each topology T_i in Figure 3 all minimal possibilities of making the topology weaker, i.e., to generate a topology T' such that $T' \sqsubset T_i$ and $\neg\exists T_M.\ T' \sqsubset T_M \sqsubset T_i$. We then argue that any such topology T' cannot satisfy *TimelyDR* for any protocol, by Lemmas 4–7. It follows by Lemma 1 that all weaker topologies $T'' \sqsubseteq T'$ also cannot satisfy the property (as this would imply that T' satisfies the property, too). Note that we can weaken a topology in three ways: by weakening the channel assumptions, by weakening the trust assumptions, or by removing a channel or a role.

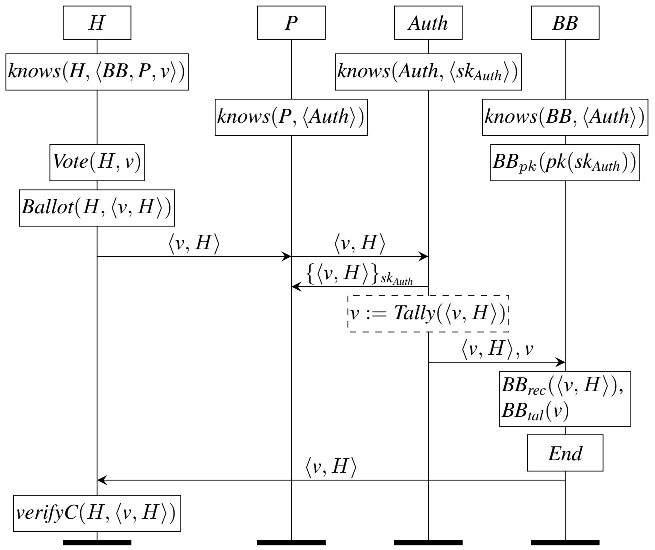
First, recall that by assumption of our protocol class, the roles *BB* and *A* and their incoming and outgoing channels are fixed. Furthermore, for in the topologies $T_1, \ldots, T_5$, we cannot remove the channels (H, P) or (P, Auth) or the roles H, P, or



H
P
$Auth$
BB
$knows(H, \langle BB, P, v\rangle)$
$knows(P, \langle Auth, sk_P\rangle)$
$knows(BB, \langle Auth\rangle)$
$Vote(H, v)$
$BB_{pkP}(pk(sk_P))$
$Ballot(H, \langle v, H\rangle)$
$\langle v, H\rangle$
$Ev(\langle v, H\rangle, \{\langle v, H\rangle\}_{sk_P})$
$\langle v, H\rangle$
$v := Tally(\langle v, H\rangle)$
$\langle v, H\rangle, v$
$BB_{rec}(\langle v, H\rangle),$
$BB_{tal}(v)$
End
$\langle v, H\rangle$
$verifyC(H, \langle v, H\rangle)$

For $b = \perp$: $Faulty(Auth, b) := \{\}$.
For $b \neq \perp$: $Faulty(Auth, b) := \{tr | \exists c, [b], pk_P.\ BB_{pkP}(pk_P) \in tr$
$\wedge Ev(b, c) \in tr \wedge ver(c, pk_P) = b \wedge BB_{rec}([b]) \in tr \wedge b \notin [b]\}$.

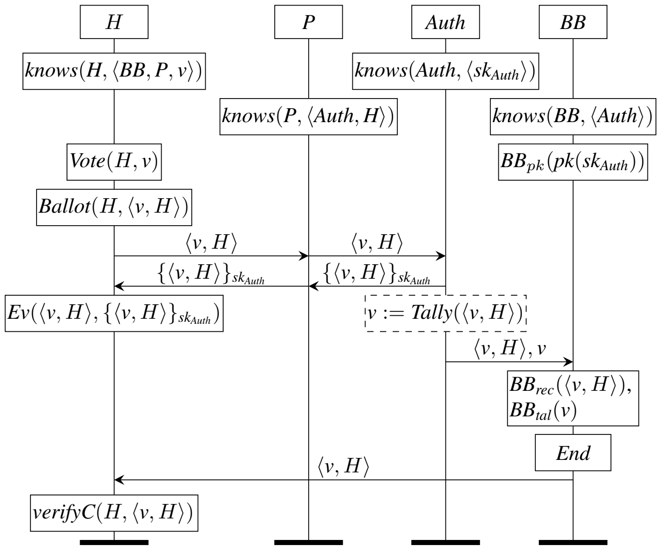
Fig. 9: The protocol Pr_3.



H
P
$Auth$
BB
$knows(H, \langle BB, P, v\rangle)$
$knows(Auth, \langle sk_{Auth}\rangle)$
$knows(P, \langle Auth\rangle)$
$knows(BB, \langle Auth\rangle)$
$Vote(H, v)$
$BB_{pk}(pk(sk_{Auth}))$
$Ballot(H, \langle v, H\rangle)$
$\langle v, H\rangle$
$\langle v, H\rangle$
$\{\langle v, H\rangle\}_{sk_{Auth}}$
$v := Tally(\langle v, H\rangle)$
$\langle v, H\rangle, v$
$BB_{rec}(\langle v, H\rangle),$
$BB_{tal}(v)$
End
$\langle v, H\rangle$
$verifyC(H, \langle v, H\rangle)$

For $b = \perp$: $Faulty(Auth, b) := \{\}$.
For $b \neq \perp$: $Faulty(Auth, b) := \{tr | \exists c, P, pk_{Auth}, [b].\ BB_{pk}(pk_{Auth}) \in tr$
$\wedge Pub(Auth, P, c) \in tr \wedge ver(c, pk_{Auth}) = b \wedge BB_{rec}([b]) \in tr \wedge b \notin [b]\}$.

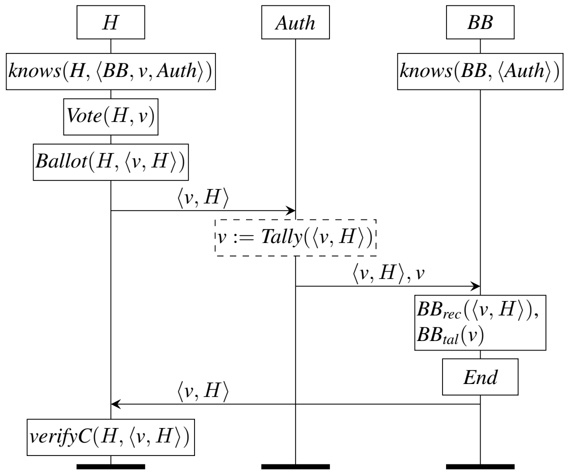
Fig. 11: The protocol Pr_5.



H
P
$Auth$
BB
$knows(H, \langle BB, P, v\rangle)$
$knows(Auth, \langle sk_{Auth}\rangle)$
$knows(P, \langle Auth, H\rangle)$
$knows(BB, \langle Auth\rangle)$
$Vote(H, v)$
$BB_{pk}(pk(sk_{Auth}))$
$Ballot(H, \langle v, H\rangle)$
$\langle v, H\rangle$
$\langle v, H\rangle$
$\{\langle v, H\rangle\}_{sk_{Auth}}$
$\{\langle v, H\rangle\}_{sk_{Auth}}$
$Ev(\langle v, H\rangle, \{\langle v, H\rangle\}_{sk_{Auth}})$
$v := Tally(\langle v, H\rangle)$
$\langle v, H\rangle, v$
$BB_{rec}(\langle v, H\rangle),$
$BB_{tal}(v)$
End
$\langle v, H\rangle$
$verifyC(H, \langle v, H\rangle)$

For $b = \perp$: $Faulty(Auth, b) := \{\}$.
For $b \neq \perp$: $Faulty(Auth, b) := \{tr | \exists c, pk_{Auth}, [b].\ BB_{pk}(pk_{Auth}) \in tr$
$\wedge Ev(b, c) \in tr \wedge ver(c, pk_{Auth}) = b \wedge BB_{rec}([b]) \in tr \wedge b \notin [b]\}$.

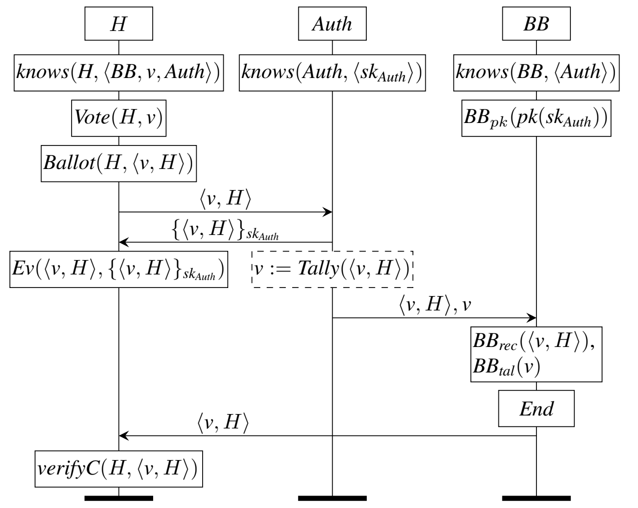
Fig. 10: The protocol Pr_4.



H
$Auth$
BB
$knows(H, \langle BB, v, Auth\rangle)$
$knows(BB, \langle Auth\rangle)$
$Vote(H, v)$
$Ballot(H, \langle v, H\rangle)$
$\langle v, H\rangle$
$v := Tally(\langle v, H\rangle)$
$\langle v, H\rangle, v$
$BB_{rec}(\langle v, H\rangle),$
$BB_{tal}(v)$
End
$\langle v, H\rangle$
$verifyC(H, \langle v, H\rangle)$

$Faulty(Auth, b) := \{tr | \exists H, [b], v.\ Pub(H, Auth, b) \in tr$
$\wedge BB_{rec}([b]) \in tr \wedge b \notin [b] \wedge b = \langle v, H\rangle\}$.

Fig. 12: The protocol Pr_6.



H
$Auth$
BB
$knows(H, \langle BB, v, Auth\rangle)$
$knows(Auth, \langle sk_{Auth}\rangle)$
$knows(BB, \langle Auth\rangle)$
$Vote(H, v)$
$BB_{pk}(pk(sk_{Auth}))$
$Ballot(H, \langle v, H\rangle)$
$\langle v, H\rangle$
$\{\langle v, H\rangle\}_{sk_{Auth}}$
$Ev(\langle v, H\rangle, \{\langle v, H\rangle\}_{sk_{Auth}})$
$v := Tally(\langle v, H\rangle)$
$\langle v, H\rangle, v$
$BB_{rec}(\langle v, H\rangle),$
$BB_{tal}(v)$
End
$\langle v, H\rangle$
$verifyC(H, \langle v, H\rangle)$

For $b = \perp$: $Faulty(Auth, b) := \{\}$.
For $b \neq \perp$: $Faulty(Auth, b) := \{tr | \exists c, pk_{Auth}, [b].\ BB_{pk}(pk_{Auth}) \in tr$
$\wedge Ev(b, c) \in tr \wedge ver(c, pk_{Auth}) = b \wedge BB_{rec}([b]) \in tr \wedge b \notin [b]\}$

Fig. 13: The protocol Pr_7.

Auth, as otherwise the functional property cannot hold. For the same reason, we cannot remove the channel $(H, Auth)$ or the roles H or $Auth$ in the topologies T_6 and T_7. We further argue that we cannot weaken any other assumptions either.

Topology T_1: When generating T' by making the channel (H, P) default or by making P untrusted, T' cannot satisfy *TimelyDR* by Lemma 4. Also, by Lemma 5, the same is true when we make the channel $(P, Auth)$ reliable only.

Topology T_2: When generating T' by making the channel $(P, Auth)$ default or by making P untrusted, T' cannot satisfy *TimelyDR* by Lemma 4. Also, by Lemma 5, the same is true when we make the channel (H, P) reliable only.

Topology T_3: When generating T' by making any of the channels (H, P) or $(P, Auth)$ default, T' cannot satisfy *TimelyDR* by Lemma 4. Also, by Lemma 5, the same is true when we make P of type *trustFwd* only.

Topology T_4: When generating T' by making the channels (H, P) or $(P, Auth)$ default or by making P untrusted, T' cannot satisfy *TimelyDR* by Lemma 4. Also, by Lemma 5, the same is true when we make *Auth* untrusted, or when we make either of the channels $(Auth, P)$ or (P, H) default.

Topology T_5: When generating T' by making the channels (H, P) or $(P, Auth)$ default or by making P untrusted, T' cannot satisfy *TimelyDR* by Lemma 4. Also, by Lemma 5, the same is true when we make *Auth* untrusted, or when we make the channel $(Auth, P)$ reliable only.

Topology T_6: When generating T' by making the channel $(H, Auth)$ reliable, T' cannot satisfy *TimelyDR* by Lemma 7.

Topology T_7: When generating T' by making the channel $(H, Auth)$ default, T' cannot satisfy *TimelyDR* by Lemma 6. Also, by Lemma 7, the same is true when we make *Auth* untrusted, or when we make the channel $(Auth, H)$ default.

In all cases, there are no further options to weaken the given topology in a minimal way. □

Finally, we use the above results to prove Theorem 1.

Proof of Theorem 1. The theorem's statement is a direct consequence of Lemmas 1 and 8 and because $T_1, \dots, T_7$ are the only minimal topologies that satisfy all requirements necessary by Lemmas 4–7. The latter is established by an exhaustive case distinction on the finitely many possible topologies: each possible way of exchanging one assumption in a topology $T_1, \dots, T_7$ by another one, results either in a topology T' that does not satisfy the conditions required by Lemmas 4–7 or in a topology T' that makes stronger assumptions than one of the seven topologies and is thus not minimal, i.e., $\exists T_j \in \{T_1, \dots, T_7\}.\ T_j \sqsubset T'$. □

B. Proof of Theorem 2

We next prove Theorem 2 from Section VI, which states that *Uniqueness*$(Auth)$ can be used to infer *VoterA*$(Auth)$ in disputes *D2* in protocols that satisfy certain conditions.

Proof of Theorem 2. Let *Pr* be a protocol as assumed by the theorem. Let tr be a trace in $TR(Pr, T)$ such that $tr \in Uniqueness(Auth)$ and such that there is a voter H for which $verifyA(H, \emptyset) \in tr$. All traces where no such signal *verifyA* is recorded trivially satisfy *VoterA*$(Auth)$ and as *Pr* is a protocol without re-voting, *verifyA* is only recorded for honest voters who abstain (that is, there are no signals $verifyA(H, x)$ for $x \neq \emptyset$).

We make a case distinction. First, assume

$$\neg(\exists [b], b.\ BB_{rec}([b]) \in tr \wedge b \in [b] \wedge castBy(b) = H).$$

Then, *VoterA*$(Auth)$ holds by Definition 3.

Second, assume $\exists [b], b.\ BB_{rec}([b]) \in tr \wedge b \in [b] \wedge castBy(b) = H$. Thus, we consider a trace where

$$\begin{aligned} & tr \in Uniqueness(Auth) \wedge verifyA(H, \emptyset) \in tr \\ & \qquad \wedge BB_{rec}([b]) \in tr \wedge b \in [b] \wedge castBy(b) = H \\ \overset{(1)}{\Longrightarrow} & tr \in Uniqueness(Auth) \wedge BB_{rec}([b]) \in tr \\ & \wedge b \in [b] \wedge castBy(b) = H \wedge \neg\exists m, A.\ send(H, A, m) \in tr \\ \overset{(2)}{\Longrightarrow} & \neg\exists b'. b' \neq \perp \wedge tr \notin Faulty(Auth, b') \\ \overset{(3)}{\Longrightarrow} & \exists b'. tr \in Faulty(Auth, b'). \end{aligned}$$

In (1), we use that *verifyA* is only recorded for an honest voter who abstains and thus, by assumption, does not send any message. In (2) we use that *Uniqueness*$(Auth)$ holds and therefore there cannot be any $b' \neq \perp$ such that $tr \notin Faulty(Auth, b')$ as this would require that $send(H, A', m')$ is in the trace for some A' and m' such that $m' \vdash b$, which is false. Thus, for all ballots $b' \neq \perp$ the trace tr must be in $Faulty(Auth, b')$ and Step (3) follows (for an arbitrary choice of a ballot $b' \neq \perp$). Thus, tr satisfies *VoterA*$(Auth)$ by Definition 3. □

C. Analyzing a mixnet-based voting protocol with dispute resolution

We present the details of the protocol *MixVote* and show that it satisfies dispute resolution as well as other standard voting properties. To formalize the protocol, we first extend the protocol model from Section II with new functions and equations. Then, we present the communication topology and the detailed protocol specification. Finally, we present the (instantiated) dispute resolution properties, introduce new properties, and analyze the protocol. Our model and the definitions of the properties other than dispute resolution are closely related to [5], where a voting protocol has been analyzed in a formalism supported by the Tamarin tool.

1) New functions and equations: Let $[l_1]$ and $[l_2]$ be two lists. We denote by $[l_1] \subseteq_l [l_2]$ that the elements of $[l_1]$ are a sub-multiset of the elements of $[l_2]$. Next, we introduce a generalization of the signature verification function *ver* that is applied to a list of signed messages and a list of verification keys, where the lists are of the same length. The function is denoted by $ver_L([s],[k])$ and if each signed message $[s]_i$ is verified with the verification key $[k]_i$, then the function returns the list of messages that were included in $[s]$ but with the signatures removed. Otherwise, the function returns the default value $\perp$.

$$ver_L([s],[k]) := \begin{cases} [m], & \forall i.\ ver([s]_i,[k]_i) = [m]_i \wedge [m]_i \neq \perp \\ \perp, & \text{otherwise.} \end{cases}$$

MixVote uses non-interactive zero knowledge proofs to shuffle, i.e., randomly permute and re-encrypt, and decrypt ballots in a publicly verifiable manner. First, we introduce a probabilistic asymmetric encryption scheme. The encryption of the message m under the public key *pk* and using the randomness r is denoted by $\{m\}^r_{pk}$ and the decryption of the ciphertext c with the private key k by $decp(c,k)$. The functions obey the equation $decp(\{m\}^r_{pk(k)},k) = m$. Then, we introduce $zkp([i],[o],k)$ to denote the non-interactive zero knowledge proof that the list $[i]$ was correctly shuffled and its elements correctly decrypted to the elements in $[o]$. Thereby, k denotes the private key needed to decrypt the messages in $[i]$ and to generate a correct proof. A zero knowledge proof can be verified using the function $ver_{zk}(p,[i'],[o'],pk')$, which takes as input a proof p, two lists $[i']$ and $[o']$, where the latter presumably contains the decryptions of the former up to permutation, and a public key pk'. Such a verification is successful if the following three conditions hold.

1) p is a non-interactive zero knowledge proof that was constructed with respect to permutations of the two lists that were input to the verification function ($[i']$ and $[o']$ above).
2) The elements of the first list $[i']$ correspond to encryptions of the elements of the second list $[o']$, but can be permuted.
3) The proof p was constructed with the private key k corresponding to the public key $pk' = pk(k)$ used in the verification and the elements in the list $[i']$ are encrypted with the same key pk'.

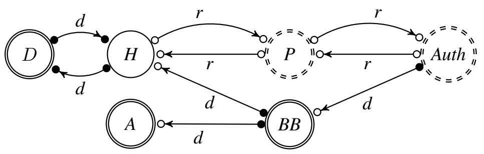


Fig. 14: The topology T_{MV} for the protocol *MixVote*.

The last condition means that only an agent who possesses the private key to decrypt the messages in $[i']$ can construct a valid proof. The following equation models these conditions, where π_1, π_2, and π_3 denote arbitrary permutations and $\pi[x]$ denotes that permutation π is a applied to the list $[x]$.

$$ver_{zk}(zkp(\pi_1[\{m\}^r_{pk(k)}],\pi_2[m],k),\pi_3[\{m\}^r_{pk(k)}],[m], pk(k)) = true.$$

These functions could, for example, be realized by the scheme of *Groth* [22], who proposes a zero knowledge proof for the combined shuffle-and-decrypt operation. In [22], several decryption servers each possess a share of the private key corresponding to the public key used for the encryptions. Each server, in turn, shuffles the ballots and decrypts them with respect to its key share. As we only consider one agent *Auth*, we can use *Groth*'s scheme where *Auth* performs all servers' computations.

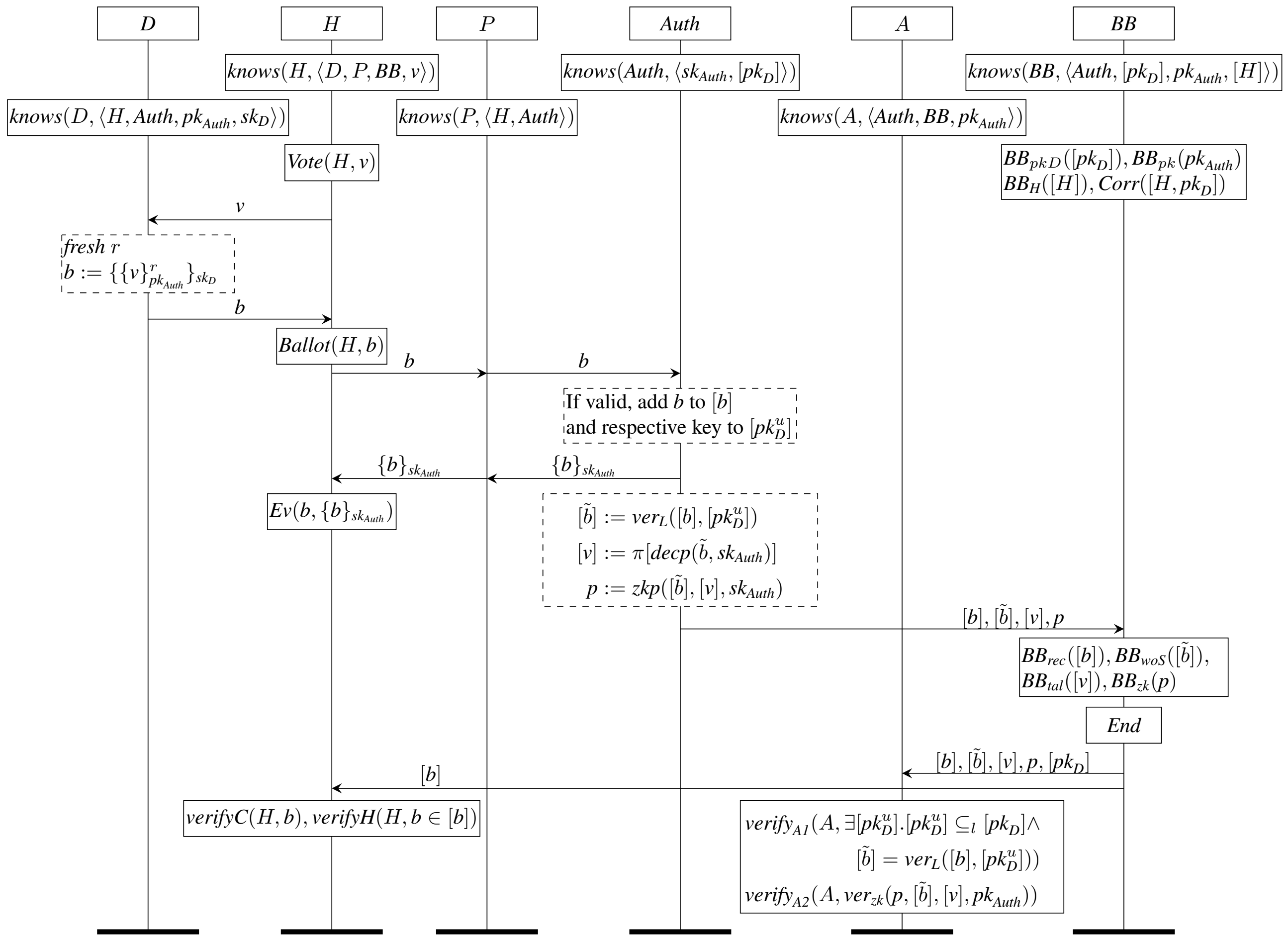
In addition to the new functions, we extend our protocol model with new signals, which we explain next while presenting the protocol.

2) Protocol and communication topology: *MixVote*'s communication topology T_{MV} and detailed protocol steps are respectively depicted in Figures 14 and 15. As explained in Section VI-D2, we present the protocol as a message sequence chart.

The topology specifies roles for the trusted off-line device D, only connected to the voter H, and the platform P, through which H can access *Auth*. Apart from the fixed assumptions about the roles *BB*, A and their incoming and outgoing channels, the topology specifies the following assumptions. We assume that the channels (H,D) and (D,H) are secure and default. Moreover, the network between H and *Auth* is insecure but the channels (H,P), $(P,Auth)$, $(Auth,P)$, and (P,H) reliably deliver messages. P is partially trusted to forward messages and *Auth* is partially trusted to reply with a confirmation when it receives a valid ballot. Except for the device D, the topology is as the topology T_4 (Figure 3d) and can be interpreted as explained in Section VI-A.

As *Auth* naturally learns how each voter voted when collecting and tallying the ballots, there are no privacy guarantees if *Auth* is not trusted. For simplicity, and as other authors [1], [5], [14], we thus model a trusted *Auth* when examining privacy properties. In reality, this trust could be distributed as is done in other work, e.g., [15], [20].

We assume that, at each point in time, only one election takes place, i.e., there are no parallel sessions. Moreover, the protocol's setup specifies that there is a single agent *Auth* and


D
H
P
$Auth$
A
BB
$knows(H, \langle D, P, BB, v\rangle)$
$knows(Auth, \langle sk_{Auth}, [pk_D]\rangle)$
$knows(BB, \langle Auth, [pk_D], pk_{Auth}, [H]\rangle)$
$knows(D, \langle H, Auth, pk_{Auth}, sk_D\rangle)$
$knows(P, \langle H, Auth\rangle)$
$knows(A, \langle Auth, BB, pk_{Auth}\rangle)$
$Vote(H, v)$
$BB_{pkD}([pk_D]), BB_{pk}(pk_{Auth})$
$BB_H([H]), Corr([H, pk_D])$
v
$fresh\ r$
$b := \{\{v\}^r_{pk_{Auth}}\}_{sk_D}$
b
$Ballot(H, b)$
b
b
If valid, add b to $[b]$
and respective key to $[pk_D^u]$
$\{b\}_{sk_{Auth}}$
$\{b\}_{sk_{Auth}}$
$Ev(b, \{b\}_{sk_{Auth}})$
$[\tilde{b}] := ver_L([b], [pk_D^u])$
$[v] := \pi[decp(\tilde{b}, sk_{Auth})]$
$p := zkp([\tilde{b}], [v], sk_{Auth})$
$[b], [\tilde{b}], [v], p$
$BB_{rec}([b]), BB_{woS}([\tilde{b}]),$
$BB_{tal}([v]), BB_{zk}(p)$
End
$[b], [\tilde{b}], [v], p, [pk_D]$
$[b]$
$verifyC(H, b), verifyH(H, b \in [b])$
$verify_{A1}(A, \exists [pk_D^u].[pk_D^u] \subseteq_l [pk_D] \wedge$
$[\tilde{b}] = ver_L([b], [pk_D^u]))$
$verify_{A2}(A, ver_{zk}(p, [\tilde{b}], [v], pk_{Auth}))$

For $b = \perp$: $Faulty(Auth, b) := \{\}$.

For $b \neq \perp$: $Faulty(Auth, b) := \{tr | (\exists [b], pk_{Auth}, c.\ BB_{pk}(pk_{Auth}) \in tr \wedge Ev(b, c) \in tr \wedge ver(c, pk_{Auth}) = b \wedge BB_{rec}([b]) \in tr \wedge b \notin [b])$
$\vee (\exists [b], [pk_D], [\tilde{b}].\ BB_{rec}([b]) \in tr \wedge BB_{pkD}([pk_D]) \in tr \wedge BB_{woS}([\tilde{b}]) \in tr \wedge (\neg \exists [pk_D^u].\ [pk_D^u] \subseteq_l [pk_D] \wedge [\tilde{b}] = ver_L([b], [pk_D^u]))))\}$.

Fig. 15: The protocol *MixVote*, or Pr_{MV}, where $pk_D = pk(sk_D)$, $pk_{Auth} = pk(sk_{Auth})$, and $castBy(b) = H$ holds iff $\exists pk.\ ver(b, pk) \neq \perp \wedge \langle H, pk\rangle \in [H, pk] \wedge Corr([H, pk]) \in tr$. The protocol's setup specifies a single agent *Auth*, that each voter H is associated with a unique trusted off-line device D, and that there is no restriction on the relation between voters H and platforms P, i.e., several voters can be connected with the same P. The role for a voter H who abstains consists of receiving the list of recorded ballots from the bulletin board followed by the signal $verifyA(H, \emptyset)$.

that each voter H has a unique trusted device D to which he has exclusive access. In contrast, there is no fixed relation between platforms and voters, i.e., several voters may use the same platform P to cast their ballot. For readability, we explain the protocol for one voter, device, and platform in Figure 15. At the protocol's start, the device and authority each know a unique private key (sk_D and sk_{Auth}, respectively) and know each others corresponding public key, which is also published on the bulletin board *BB*. Moreover, *BB* contains the list of all eligible voters $[H]$ and information on which signing key corresponds to which eligible voter, i.e., which signing key is installed on that voter's device. We denote the later by the signal $Corr([H, pk_D])$, where each pair $\langle H, pk_D\rangle$ in the list denotes that the signing key corresponding to pk_D is installed on H's device.

To compute a ballot, a voter enters his vote on his device D. D generates a fresh random number r (denoted by *fresh*), uses r to probabilistically encrypt the vote under *Auth*'s public key, and signs the resulting encryption with its signing key. D displays the computed ballot to H and H casts the ballot (denoted by the signal *Ballot*) by entering it on his platform P from where it is forwarded to *Auth*. This communication could, for example, be realized by D displaying the ballot as a QR code and H scanning this code with P. Upon receiving a ballot, *Auth* checks whether it is well-formed and signed with an eligible voter's key. If this is the case, *Auth* adds the ballot to the list of recorded ballots $[b]$ and the respective verification key to $[pk_D^u]$ to keep track of the used public keys, as each voter can only vote once. Also, *Auth* sends as confirmation the signed ballot back to H via P, where the voter keeps it as evidence in case of subsequent disputes.

After the voting phase ends, *Auth* computes the tally. Instead of the function *Tally*, we describe this procedure in terms of several, more detailed steps: *Auth* removes the signatures from the recorded ballots, shuffles and decrypts the resulting encryptions with its private key, and produces a zero knowledge proof that these operations were done correctly. *Auth* then publishes all the lists and the proof on the bulletin board, where an auditor can read them.

The auditor first checks that all recorded ballots contain a signature corresponding to a unique public key in $[pk_D]$. Then, the auditor verifies the zero knowledge proof. We denote this respectively by the explicit signals $\mathit{verify}_{A1}(A, p_1)$ and $\mathit{verify}_{A2}(A, p_2)$, which record that the agent A checks whether the predicates p_1 and p_2 hold. In the protocol's traces, these signals are recorded as $\mathit{verify}_{A1}(A, p_1, t_1)$ and $\mathit{verify}_{A2}(A, p_2, t_2)$, with $t_1, t_2 \in \{\mathit{true}, \mathit{false}\}$ respectively indicating whether the predicate p_1 and p_2 is satisfied in this particular trace. This allows us to refer both to the terms that are evaluated in the predicate and the predicate's truth value during execution.

Moreover, the voter reads the list of recorded ballots from *BB* and checks whether his ballot is contained in this list. In addition to $\mathit{verifyC}(H, b)$, this is recorded by the signal $\mathit{verifyH}(H, b \in [b])$, which we will need to express individual verifiability. As with verify_{A1} and verify_{A2}, *verifyH* denotes the agent and the checked predicate and contains in the protocol's trace a third argument stating whether the predicate is satisfied.

Finally, a voter who abstains does not send any messages. After the results are published, he reads from the bulletin board the list of recorded ballots and believes at that step that no ballot is recorded for him, which is recorded by the signal $\mathit{verifyA}(H, \emptyset)$.

We complete the protocol specification as follows. A ballot is considered to be cast by a voter when the ballot's signature can be verified with the public key associated with this voter. That is $\mathit{castBy}(b) = H$ iff there exists a verification key *pk* such that $\mathit{ver}(b, pk) \neq \perp \wedge \langle H, pk\rangle \in [H, pk] \wedge \mathit{Corr}([H, pk]) \in tr$. Moreover, we define the verdict that *Auth* behaved dishonestly as the set of traces where a) there exists evidence consisting of a ballot signed by *Auth* but this ballot is not included in the recorded ballots on the bulletin board or b) there are recorded ballots that are not signed by a unique eligible voter. The corresponding set *Faulty* is defined in Figure 15.

3) Security properties: With the above instantiation of the function *Faulty*, all introduced dispute resolution properties are now defined and can be analyzed. As the definition of *Uniqueness*(*Auth*) additionally contains *castBy*, we explicitly instantiate this part for simplicity.

Definition 9. Let the length of the list $[b]$ be n.

$$\begin{aligned}
\mathit{Uniqueness}(\mathit{Auth}) :=\ &\{tr \mid b \neq \perp \wedge tr \notin \mathit{Faulty}(\mathit{Auth}, b)\\
&\wedge BB_{rec}([b]) \in tr \wedge i \in \{1, \ldots, n\} \wedge j \in \{1, \ldots, n\} \implies\\
&\exists [H], i', j', [pk], pk_1, pk_2, A_1, A_2, m_1, m_2.\ BB_H([H]) \in tr\\
&\wedge \mathit{Corr}([H, pk]) \in tr \wedge\ \mathit{ver}([b]_i, pk_1) \neq \perp \wedge\ \mathit{ver}([b]_j, pk_2) \neq \perp\\
&\wedge \langle [H]_{i'}, pk_1\rangle \in [H, pk] \wedge \langle [H]_{j'}, pk_2\rangle \in [H, pk]\\
&\wedge \mathit{send}([H]_{i'}, A_1, m_1) \in tr \wedge \mathit{send}([H]_{j'}, A_2, m_2) \in tr\\
&\wedge m_1 \vdash [b]_i \wedge m_2 \vdash [b]_j \wedge (i \neq j \implies [H]_{i'} \neq [H]_{j'})\}.
\end{aligned}$$

Compared to the more general definition of *Uniqueness*(*Auth*) from Definition 7, the function *castBy* is instantiated in the fourth and fifth line.

In addition to the dispute resolution properties, we analyze standard verifiability and privacy properties that we introduce next. We start with individual verifiability, or *IndivVerif* for short, that states that whenever a voter verifies that his ballot is in the list of recorded ballots, then indeed one of the recorded ballots corresponds to his vote. The property's definition is based on individual verifiability due to [5] and [26].

Definition 10.

$$\begin{aligned}
\mathit{IndivVerif} := \{&tr \mid \mathit{verifyH}(H, b \in [b], \mathit{true}) \in tr \wedge\\
&\mathit{Vote}(H, v) \in tr \implies \exists [b'], pk_{Auth}, r, sk_D.\\
&BB_{rec}([b']) \in tr \wedge b \in [b'] \wedge b = \{\{v\}^r_{pk_{Auth}}\}_{sk_D}\}.
\end{aligned}$$

Next, we introduce the universal verifiability property *Tallied-as-recorded* that states that any auditor can verify that all recorded ballots are counted correctly in the final tally. In particular, if an auditor performs its specified checks on some lists, then the bulletin board contains the same lists of ballots, votes, and verification keys, the votes in the final tally correspond to the votes encrypted in the recorded ballots, and each ballot contains a signature associated with a different verification key in $[pk_D]$. The order of the votes and ballots can be permuted, as indicated by the permutation π.

Definition 11.

$$\begin{aligned}
\mathit{Tallied\text{-}as\text{-}Recorded} := \{&tr \mid \mathit{verify}_{A1}(A, \exists [pk_D^u].\\
&[pk_D^u] \subseteq_l [pk_D] \wedge [\tilde{b}] = \mathit{ver}_L([b], [pk_D^u]), \mathit{true}) \in tr\\
&\wedge \mathit{verify}_{A2}(A, \mathit{ver}_{zk}(p, [\tilde{b}], [v], pk_{Auth}), \mathit{true}) \in tr \implies\\
&\exists [r], [sk_D], \pi.\ BB_{rec}([b]) \in tr \wedge BB_{tal}([v]) \in tr \wedge\\
&BB_{pkD}([pk_D]) \in tr \wedge [b] = \pi[\{\{v\}^r_{pk_{Auth}}\}_{sk_D}]\\
&\wedge [pk(sk_D)] = [pk_D^u] \wedge [pk_D^u] \subseteq_l [pk_D]\}.
\end{aligned}$$

We then define *end-to-end verifiability*, or *End2EndVerif* for short, as the conjunction of individual verifiability and tallied-as-recorded.

Definition 12.

$$\mathit{End2EndVerif} := \mathit{IndivVerif} \cap \mathit{Tallied\text{-}as\text{-}Recorded}.$$

Eligibility verifiability is another universal verifiability property that is defined by *Kremer et al.* [26] as the property that *"anyone can check that each vote in the election outcome was cast by a registered voter and there is at most one vote per voter."* We denote the property by *EligVerif* and define it conditional on an auditor A performing all of its specified verifiability checks. If these checks are verified, then each vote in the published final tally must have been sent by an eligible voter H. Furthermore, no two votes in the final tally correspond to the same voter.

Definition 13. Let n be the length of the list $[v]$.

$$\begin{aligned} \mathit{EligVerif} := \{ tr \mid\ & \mathit{verify}_{A1}(A, \exists [pk_D^u].[pk_D^u] \subseteq_l [pk_D] \\ & \wedge [\tilde{b}] = \mathit{ver}_L([b], [pk_D^u]), \mathit{true}) \in tr \\ & \wedge \mathit{verify}_{A2}(A, \mathit{ver}_{zk}(p, [\tilde{b}], [v], pk_{\mathit{Auth}}), \mathit{true}) \in tr \\ & \wedge i \in \{1, \ldots, n\} \wedge j \in \{1, \ldots, n\} \\ \implies & \exists [H], i', j', A, A'.\ BB_H([H]) \in tr \wedge BB_{tal}([v]) \in tr \\ & \wedge \mathit{send}([H]_{i'}, A, [v]_i) \in tr \wedge \mathit{send}([H]_{j'}, A', [v]_j) \in tr \\ & \wedge (i \neq j \implies [H]_{i'} \neq [H]_{j'}) \}. \end{aligned}$$

Note that *Uniqueness* and *EligVerif* make similar guarantees in that both state that each element of a list is associated with a *unique eligible voter*. *EligVerif* states this guarantee with respect to all votes in the final tally. In contrast, *Uniqueness* states the guarantee for the *recorded* ballots as these are the relevant terms for the disputes.

Next, we introduce the privacy property *receipt-freeness*. Whereas *vote privacy* denotes that an adversary cannot link voters to their votes, receipt-freeness is strictly stronger and additionally requires that this holds even when a voter reveals his secrets to the adversary, i.e., when he tries to provide a receipt of his vote. This property is important, as a protocol should ensure privacy even when honest voters are forced by an adversary to reveal private information.

As [5] and similarly to [18], we define receipt-freeness as an observational equivalence property, which states that an adversary cannot distinguish between two systems. Concretely, we define a *left system* where the voter A votes v_1 and the voter B votes v_2 and a *right system* where A votes v_2 and B votes v_1. Moreover, we change the original protocol *Pr* to *Pr′* where A sends all his secrets to the adversary except that A always claims that his vote is v_1 (which is only true in the left system). We then define a set $\mathcal{S}$ of trace pairs (tr_L, tr_R) where tr_L is from the left system and tr_R from the right system. A protocol *Pr* satisfies receipt-freeness if for all traces of *Pr′* in one system, there exists a trace in the other system such that the pair of traces is contained in $\mathcal{S}$.

We use the notation $Pr_{m_1 \leftarrow m_1', m_2 \leftarrow m_2'}$ to denote the specification of the protocol *Pr* where each occurrence of the terms m_1 and m_2 is replaced by m_1' and m_2', respectively. Furthermore, we write $t_1 \approx t_2$ to denote that two traces are indistinguishable for an adversary. We use this definition informally here and refer to [3] for a formal definition of $\approx$.

Definition 14. Let *Pr′* be the protocol obtained from *Pr* as described above, let v_A and v_B be the term that denotes A's and B's vote, respectively, and let v_1 and v_2 be message terms. Receipt-freeness of the protocol *Pr* run in the topology T is defined as follows.

$$\begin{aligned} &\mathit{Receipt\text{-}freeness} := \{(tr_L, tr_R) \in \\ &\mathit{TR}(Pr'_{v_A \leftarrow v_1, v_B \leftarrow v_2}, T) \times \mathit{TR}(Pr'_{v_A \leftarrow v_2, v_B \leftarrow v_1}, T) \\ &\qquad\qquad\qquad \mid tr_L \approx tr_R\}. \end{aligned}$$

This set defines all indistinguishable trace pairs such that the traces are from two systems where A and B vote the opposite way. A reveals all secrets, except he claims in both systems that he votes v_1, which is only true in the left system.

4) Analysis: As mentioned in Section VI-D, by Theorem 1 it is possible to achieve *TimelyDR* in the topology T_{MV} as $T_4 \sqsubseteq T_{MV}$. We next analyze whether *MixVote* in Figure 15, Pr_{MV} for short, indeed satisfies all above properties when run in the topology T_{MV}. Recall that *VoterC*(*Auth*), *TimelyP*(*Auth*), and *VoterA*(*Auth*) are guarantees for the voter and should hold for a distinguished voter H, even when the authority *Auth* and all other voters are dishonest. Similarly, *Uniqueness*(*Auth*) and the verifiability properties must hold even when the authority is dishonest. In contrast, *AuthP*(*Auth*) is a guarantee for *Auth* and should hold even when all voters are dishonest. Finally, recall that we assume for receipt-freeness that *Auth* is honest.

Theorem 3. *MixVote* satisfies *VoterC*(*Auth*), *TimelyP*(*Auth*), *VoterA*(*Auth*), *AuthP*(*Auth*), *Uniqueness*(*Auth*), *IndivVerif*, *Tallied-as-Recorded*, *End2EndVerif*, *EligVerif*, and *Receipt-freeness* when run in the topology $T_{MV}^{Auth^+ H^+}$. When run in the topology $T_{MV}^{H^+}$, the protocol satisfies the same properties, except for *AuthP*(*Auth*) and *Receipt-freeness*. When run in the topology $T_{MV}^{Auth^+}$ the protocol satisfies *AuthP*(*Auth*).

Proof. To prove the theorem, we combine proofs by Tamarin with pen-and-paper proofs. We establish all properties except for *VoterA*(*Auth*), *End2EndVerif*, and *Receipt-freeness* for one voter who casts a vote in Tamarin and model each of the topologies $T_{MV}^{Auth^+ H^+}$, $T_{MV}^{H^+}$, and $T_{MV}^{Auth^+}$ in a separate Tamarin theory. Moreover, to prove *Receipt-freeness* we model two voters as explained above and automatically prove the property in the topology $T_{MV}^{Auth^+ H^+}$ using Tamarin's built in support for observational equivalence [3]. All Tamarin files can be found in [35].

As Tamarin requires specifying a fixed number of voters, we prove by hand in the Lemmas 9–15 below that the properties *TimelyP*(*Auth*), *VoterC*(*Auth*), *AuthP*(*Auth*), *IndivVerif*, *Tallied-as-Recorded*, *Uniqueness*(*Auth*), and *EligVerif* also hold for an arbitrary number of voters in the topologies claimed by the theorem. We then show in Lemma 16 that *VoterA*(*Auth*) is implied by *Uniqueness*(*Auth*) in the required topologies, as Theorem 2 can be applied. Finally, as *End2EndVerif* is the intersection of *IndivVerif* and *Tallied-as-Recorded*, it directly follows that *End2EndVerif* also holds in the required adversary models. ☐

Lemma 9. *MixVote* satisfies *TimelyP*(*Auth*) when run in the topologies $T_{MV}^{Auth^+ H^+}$ and $T_{MV}^{H^+}$.

Proof. Let H be the distinguished voter for which the property should hold, independently of whether or not there are other (dishonest) voters in the system. Let tr be a trace in $TR(Pr_{MV}, T_{MV}^{Auth^+H^+}) \cup TR(Pr_{MV}, T_{MV}^{H^+})$ such that $tr = tr' \cdot tr''$ and $Ballot(H,b) \in tr'$ and $End \in tr''$.

$$tr = tr' \cdot tr'' \wedge Ballot(H,b) \in tr' \wedge End \in tr''$$

$$\overset{(1)}{\Longrightarrow} \exists P, [b].\ send(H,P,b) \in tr \wedge BB_{rec}([b]) \in tr'$$

$$\overset{(2)}{\Longrightarrow} send(H,P,b) \in tr \wedge rec(H,P,b) \in tr \wedge send(P,Auth,b) \in tr$$

$$\overset{(3)}{\Longrightarrow} send(H,P,b) \in tr \wedge rec(P,Auth,b) \in tr$$

$$\overset{(4)}{\Longrightarrow} \exists sk_{Auth}.\ send(H,P,b) \in tr \wedge send(Auth,P,\{b\}_{sk_{Auth}}) \in tr \wedge BB_{pk}(pk(sk_{Auth})) \in tr$$

$$\overset{(5)}{\Longrightarrow} send(H,P,b) \in tr \wedge rec(Auth,P,\{b\}_{sk_{Auth}}) \in tr \wedge send(P,H,\{b\}_{sk_{Auth}}) \in tr \wedge rec(P,H,\{b\}_{sk_{Auth}}) \in tr$$

$$\overset{(6)}{\Longrightarrow} Ev(b,\{b\}_{sk_{Auth}}) \in tr$$

The first conjunct of Step (1) holds as $Ballot(H,b)$ is recorded when H sends the ballot. Note that the honest voter H only sends this message on the channel (H,P). The second part of Step (1) holds as *End* is be definition recorded after the last message is published on the bulletin board. Thus, for some list $[b]$, $BB_{rec}([b])$ is recorded in the trace before *End*. Step (2) follows by the fact that in both topologies $T_{MV}^{Auth^+H^+}$ and $T_{MV}^{H^+}$, it holds that $c(H,P) = \circ\xrightarrow{r}\circ$, $t(H) = trusted$, and $t(P) = trustFwd$. Thus, by the restrictions from Section II-E2, each message that is sent from H to P will be received by P.

Moreover, by the fact that $t(P) = trustFwd$ P always forwards messages correctly, i.e., P forwards the ballot to the (unique) authority *Auth*. Similarly to Step (2), Steps (3) and (4) follow by the assumption that the channel from P to *Auth* is reliable and that $t(Auth) = trustRpl$. In particular, as *Auth* is partially trusted to answer with the (correct) confirmation for a valid ballot, it sends back the ballot signed with its (correct) signing key (as H and his device D are honest, the ballot b is valid). By the shared initial knowledge, the setup assumptions, and since the bulletin board is honest by the topology T_{MV}, the public key corresponding to *Auth*'s signing key is published on the bulletin board (second part in Step (4)). In Step (5), we use the same reasoning as in Steps (2) and (3) for the channels $(Auth,P)$ and (P,H) which are also reliably by T_{MV}. By the protocol specification, when the honest H receives the confirmation, the signal *Ev* is recorded, where the first argument is the ballot that H sent and the second the received confirmation. Thus (6) holds.

Finally, we make a case distinction. First, let $b \in [b]$. Then, it immediately follows that *TimelyP*(*Auth*) holds. Second, let $b \notin [b]$. Then, it holds that $BB_{pk}(pk(sk_{Auth})) \in tr \wedge Ev(b,\{b\}_{sk_{Auth}}) \in tr \wedge ver(\{b\}_{sk_{Auth}}, pk(sk_{Auth})) = b \wedge BB_{rec}([b]) \in tr \wedge b \notin [b]$ and thus, by the definition of *Faulty* it holds that $tr \in Faulty(Auth,b)$. Therefore, *TimelyP*(*Auth*) also holds in this case. □

Lemma 10. *MixVote* satisfies *VoterC*(*Auth*) when run in the topologies $T_{MV}^{Auth^+H^+}$ and $T_{MV}^{H^+}$.

Proof. Let H be the distinguished voter for which the property should hold, independently of whether there are other (dishonest) voters in the system. Let tr be a trace in $TR(Pr_{MV}, T_{MV}^{Auth^+H^+}) \cup TR(Pr_{MV}, T_{MV}^{H^+})$ such that $verifyC(H,b) \in tr$.

$$verifyC(H,b) \in tr$$

$$\overset{(1)}{\Longrightarrow} \exists P, [b], BB.\ send(H,P,b) \in tr \wedge rec(BB,H,[b]) \in tr$$

$$\overset{(2)}{\Longrightarrow} send(H,P,b) \in tr \wedge send(BB,H,[b]) \in tr$$

$$\overset{(3)}{\Longrightarrow} send(H,P,b) \in tr \wedge BB_{rec}([b]) \in tr$$

As H is honest in $T_{MV}^{Auth^+H^+}$ and $T_{MV}^{H^+}$, he follows his role specification. Thus, when $verifyC(H,b)$ is in the trace, so are the signals $send(H,P,b)$ for some P and $rec(BB,H,[b])$ for some *BB* and $[b]$ as these signals are preceding the verifiability check in H's role (Step (1)). By the topology T_{MV}, the channel (BB,H) is authentic. Therefore, *BB* must have sent the message $[b]$ for H to receive it on this channel (Step (2)). Moreover, *BB* is honest by the topology T_{MV}. Thus, *BB* only sends one message containing a single list, which was previously recorded in the signal BB_{rec}. In particular, as there are no parallel sessions, this list cannot be confused with another list sent by *BB* and it follows in Step (3) that $BB_{rec}([b]) \in tr$.

We now have a trace with the same signals as in the proof of Lemma 9 after Step (1). We can thus apply the same steps as in the Proof of Lemma 9 to conclude that *VoterC*(*Auth*) holds in all the required topologies. (Note that in the proof of Lemma 9, BB_{rec} is in the subtrace tr' rather than in tr, which is however not needed in the further proof steps.) □

Lemma 11. *MixVote* satisfies *AuthP*(*Auth*) when run in the topologies $T_{MV}^{Auth^+H^+}$ and $T_{MV}^{Auth^+}$.

Proof. Let tr be a trace in $TR(Pr_{MV}, T_{MV}^{Auth^+H^+}) \cup TR(Pr_{MV}, T_{MV}^{Auth^+})$, such that $hon(Auth) \in tr$. We show that, for all ballots b, this trace is not in $Faulty(Auth,b)$. First, consider $b = \perp$. As $Faulty(Auth,\perp) = \{\}$ and as we just defined that $hon(Auth) \in tr$, it holds that $tr \notin Faulty(Auth,\perp)$.

We thus consider only ballots b different from $\perp$ in the following and show that tr is not in $Faulty(Auth,b)$ by separately showing that tr satisfies neither of the two disjuncts of the definition of $Faulty(Auth,b)$ for ballots $b \neq \perp$. We start with the second disjunct and assume that the corresponding lists are published on the bulletin board. If this is not given, the second disjunct is false and we are done.

$$BB_{rec}([b]) \in tr \wedge BB_{pkD}([pk_D]) \in tr \wedge BB_{woS}([\tilde{b}]) \in tr$$

$$\overset{(1)}{\Longrightarrow} \exists sk_{Auth} BB, [v], p.\ knows(Auth, \langle sk_{Auth}, [pk_D]\rangle) \in tr \wedge send(Auth, BB, \langle [b], [\tilde{b}], [v], p\rangle) \in tr$$

$$\overset{(2)}{\Longrightarrow} \exists [pk_D^u].\ [\tilde{b}] = ver_L([b],[pk_D^u]) \wedge [pk_D^u] \subseteq_l [pk_D]$$

The first part of Step (1) holds by the shared initial knowledge specified in the protocol and as *BB* is honest by the topology T_{MV}. Thus, *BB* and *Auth* share the same list $[pk_D]$ in their initial knowledge that *BB* has published in the signal BB_{pkD}. Also, there are no parallel sessions so no message confusion is possible. The second part of Step (1) holds as the honest *BB* only records the lists $[b]$ and $[\tilde{b}]$ after it has received them from *Auth* as the first two of four messages. As the channel $(Auth, BB)$ is authentic by topology T_{MV}, *Auth* must thus have sent these messages for *BB* to receive them.

Next, as *Auth* only sends one message to *BB* and as there are no parallel sessions, the lists sent by *Auth* respectively correspond to the correctly computed ballots, ballots without signatures, tallied votes, and the zero knowledge proof. Thus, in particular, each recorded ballot is signed by a unique eligible voter's signing key, hence Step (2) holds. (Note that the honest *Auth* uses the list $[pk_D]$ from its initial knowledge for its check whether a ballot is valid.) This is a contradiction to the second disjunct in *Faulty*'s definition, which thus cannot be satisfied by the trace tr.

Next, we show that the first disjunct in the definition of $Faulty(Auth, b)$ for ballots $b \neq \perp$ is also not satisfied by tr. Let S be an explicit signal, i.e., a top-level function in our protocol model, and let x and y be arbitrary terms. Assume that there is a signal in the trace tr, which contains a subterm signed by *Auth*'s signing key. Recall that $m_2 \vdash m_1$ denotes that m_1 is a subterm of the composed message m_2.

$$\begin{aligned} &hon(Auth) \in tr \wedge S(x) \in tr \wedge x \vdash \{y\}_{sk_{Auth}} \\ &\qquad\qquad \wedge BB_{pk}(pk(sk_{Auth})) \in tr \\ \Rightarrow &\exists A, x'.\ send(Auth, A, x') \in tr \wedge x' \vdash \{y\}_{sk_{Auth}} \end{aligned}$$

The implication holds since, by the initial knowledge specified in the protocol, only *Auth* knows the signing key sk_{Auth} corresponding to the public key published in BB_{pk} by *BB* (which is honest by topology T_{MV}). Furthermore, no one can learn sk_{Auth} during the protocol execution and no explicit signal is recorded in *Auth*'s role. In particular, the former holds as *Auth* only sends out messages containing sk_{Auth} where sk_{Auth} is used as a signing key, which cannot be extracted by our equational theory. It follows that *Auth* must have sent a term containing $\{y\}_{sk_{Auth}}$ for it to be recorded in an explicit signal in the trace and thus the implication holds.

Next, we make a case distinction. First, suppose that there are no recorded ballots on the bulletin board, that is $\neg\exists[b].\ BB_{rec}([b]) \in tr$. Then it follows that $\forall b.\ tr \notin Faulty(Auth, b)$.

Second, suppose that there are some recorded ballots on the bulletin board.

$$\begin{aligned} &\exists[b].\ send(Auth, A, x') \in tr \wedge x' \vdash \{y\}_{sk_{Auth}} \wedge \\ &\qquad\qquad BB_{rec}([b]) \in tr \\ \xrightarrow{(1)} &\exists BB, [\tilde{b}], [v], p.\ send(Auth, A, x') \in tr \wedge x' \vdash \{y\}_{sk_{Auth}} \\ &\qquad \wedge send(Auth, BB, \langle [b], [\tilde{b}], [v], p\rangle) \in tr \\ \xrightarrow{(2)} &y \in [b] \end{aligned}$$

As *BB* is honest by topology T_{MV}, it only records $BB_{rec}([b])$ if it has previously received this term from *Auth* and as the first of four terms. As by T_{MV} the channel $(Auth, BB)$ is authentic, *Auth* must have sent these terms for *BB* to receive them (Step (1)). Next, as *Auth* is honest and follows its role specification, the only messages sent by *Auth* that are signed by sk_{Auth} are the ballots, which are also all contained in the recorded ballots $[b]$, i.e., in the first list of the only message that *Auth* sends to *BB* (Step (2)). Again no message confusions are possible as *Auth* only sends one message to *BB* and as there are no parallel sessions. Finally, as the above observations hold for any explicit signal S containing *Auth*'s signature, they hold in particular for the signal $S = Ev$. Moreover, as *BB* is honest there is only one signal $BB_{pk}(pk_{Auth})$ in the trace. Thus, for any b, pk_{Auth}, c such that $BB_{pk}(pk_{Auth}) \in tr \wedge Ev(b, c) \in tr \wedge ver(c, pk_{Auth}) = b$ it follows that, for some $[b]$, $BB_{rec}([b]) \in tr \wedge b \in [b]$. Therefore, the first disjunct of the definition of $Faulty(Auth, b)$ for ballots $b \neq \perp$ is also not satisfied by tr and we conclude that for all ballots b it holds that $tr \notin Faulty(Auth, b)$. ☐

Lemma 12. *MixVote* satisfies *IndivVerif* when run in the topologies $T_{MV}^{Auth^+H^+}$ and $T_{MV}^{H^+}$.

Proof. Let H be the distinguished voter for which the property should hold, independently of whether or not there are other (dishonest) voters in the system. Let tr be a trace in $TR(Pr_{MV}, T_{MV}^{Auth^+H^+}) \cup TR(Pr_{MV}, T_{MV}^{H^+})$ such that $verifyH(H, b \in [b], true) \in tr$ and $Vote(H, v) \in tr$.

$$\begin{aligned} &verifyH(H, b \in [b], true) \in tr \wedge Vote(H, v) \in tr \\ \xrightarrow{(1)} &\exists BB, D, P.\ knows(H, \langle D, P, BB, v\rangle) \in tr \\ &\qquad \wedge rec(D, H, b) \in tr \wedge rec(BB, H, [b]) \in tr \wedge b \in [b] \\ &\qquad \wedge Vote(H, v) \in tr \\ \xrightarrow{(2)} &send(D, H, b) \in tr \wedge send(BB, H, [b]) \in tr \\ &\qquad \wedge Vote(H, v) \in tr \\ \xrightarrow{(3)} &\exists r, pk_{Auth}, sk_D, v'.\ rec(H, D, v') \in tr \wedge b = \{\{v'\}^r_{pk_{Auth}}\}_{sk_D} \\ &\qquad \wedge BB_{rec}([b]) \in tr \wedge Vote(H, v) \in tr \\ \xrightarrow{(4)} &send(H, D, v') \in tr \wedge b = \{\{v'\}^r_{pk_{Auth}}\}_{sk_D} \\ &\qquad \wedge Vote(H, v) \in tr \\ \xrightarrow{(5)} &send(H, D, v) \in tr \wedge b = \{\{v\}^r_{pk_{Auth}}\}_{sk_D}. \end{aligned}$$

By the assumption $tr \in TR(Pr_{MV}, T_{MV}^{Auth^+H^+}) \cup TR(Pr_{MV}, T_{MV}^{H^+})$, H is honest and thus only performs the check with the ballot b that he has previously received from D and the list of recorded ballots $[b]$ that he has previously received from *BB* (Step (1)). Moreover, $b \in [b]$ holds since the last argument of the verifiability check is *true*. By the topology T_{MV}, the channel (D, H) is secure and the channel (BB, H) is authentic. Therefore, D and *BB* must respectively have sent the messages b and $[b]$ for H to receive them on these channels (Step (2)).

BB is honest by the topology T_{MV}. Thus, *BB* only sends one message containing a single list which was previously recorded in the signal BB_{rec}. Moreover, as there are no parallel sessions, this list cannot be confused with another list sent by *BB*. It follows in Step (3) that $BB_{rec}([b]) \in tr$. The other parts of Step (3) follow as D is honest by the topology T_{MV}. Thus, D only sends a ballot b that contains a vote v', encrypted and signed, that it has previously received from H. Also, no message confusion is possible as there are no parallel sessions and D only sends one message in the protocol. As by the topology T_{MV} the channel (H, D) is secure, H must have sent v' for D to receive it (Step (4)). As H is honest and there are no parallel sessions, he only sends on the channel (H, D) the vote that is also recorded in the signal $Vote(H, v)$. Thus, it holds that $v = v'$ and Step (5) follows.

□

Lemma 13. *MixVote* satisfies *Tallied-as-Recorded* when run in the topologies $T_{MV}^{Auth^+H^+}$ and $T_{MV}^{H^+}$.

Proof. Let tr be a trace in $TR(Pr_{MV}, T_{MV}^{Auth^+H^+}) \cup TR(Pr_{MV}, T_{MV}^{H^+})$ such that $q_1 = verify_{A1}(A, \exists[pk_D^u].[pk_D^u] \subseteq_l [pk_D] \wedge [\tilde{b}] = ver_L([b], [pk_D^u]), true) \in tr$ and $q_2 = verify_{A2}(A, ver_{zk}(p, [\tilde{b}], [v], pk_{Auth}), true) \in tr$.

$$\begin{aligned}
& q_1 \wedge q_2 \\
\xrightarrow{(1)} & q_1 \wedge q_2 \wedge \exists[r], \pi.\ [\tilde{b}] = \pi[\{v\}^r_{pk_{Auth}}] \\
\xrightarrow{(2)} & q_1 \wedge q_2 \wedge \exists[sk_D].\ [\tilde{b}] = \pi[\{v\}^r_{pk_{Auth}}] \wedge [b] = [\{\tilde{b}\}_{sk_D}] \\
& \qquad \wedge [pk(sk_D)] = [pk_D^u] \wedge [pk_D^u] \subseteq_l [pk_D] \\
\xrightarrow{(3)} & q_1 \wedge q_2 \wedge [b] = \pi[\{\{v\}^r_{pk_{Auth}}\}_{sk_D}] \wedge [pk(sk_D)] = [pk_D^u] \\
& \qquad \wedge [pk_D^u] \subseteq_l [pk_D] \\
\xrightarrow{(4)} & \exists BB.\ rec(BB, A, \langle [b], [\tilde{b}], [v], p, [pk_D] \rangle) \in tr \\
\xrightarrow{(5)} & send(BB, A, \langle [b], [\tilde{b}], [v], p, [pk_D] \rangle) \in tr \\
\xrightarrow{(6)} & BB_{rec}([b]) \in tr \wedge BB_{tal}([v]) \in tr \wedge BB_{pkD}([pk_D]) \in tr.
\end{aligned}$$

Step (1) holds as the verification $verify_{A2}$ succeeds, indicated by its third argument *true*, and by the definition of the verification function for zero knowledge proofs ver_{zk} (see Appendix C1). As the third argument of $verify_{A1}$ is also *true* and by the definition of $\subseteq_l$ (see Appendix C1), Step (2) holds. Next, Step (3) combines the results of the first two steps. In (4), we use that the auditor A is honest by the topology T_{MV} and follows its role specification. Thus, q_1 and q_2 are only in the trace if A has previously received the corresponding lists from *BB*. As the channel (BB, A) is authentic by the topology T_{MV}, *BB* must thus have sent these messages for A to receive them on this channel (Step (5)). Finally, (6) holds as by the topology T_{MV}, *BB* is honest and thus only sends out these lists if it has previously published them, i.e., the corresponding signals were recorded in the trace. Thereby, as *BB* only sends one message of this form and as there are no parallel sessions, no message confusion is possible. Thus, we have shown that all required signals are in the trace and that the lists have the required relations. □

Lemma 14. *MixVote* satisfies *Uniqueness*(*Auth*) when run in the topologies $T_{MV}^{Auth^+H^+}$ and $T_{MV}^{H^+}$.

Proof. Let tr be a trace in $TR(Pr_{MV}, T_{MV}^{Auth^+H^+}) \cup TR(Pr_{MV}, T_{MV}^{H^+})$ such that $tr \notin Faulty(Auth, b')$, $b' \neq \perp$, and $BB_{rec}([b]) \in tr$.

$$\begin{aligned}
& b' \neq \perp \wedge tr \notin Faulty(Auth, b') \wedge BB_{rec}([b]) \in tr \\
\xrightarrow{(1)} & \exists[\tilde{b}], [pk_D], [H].\ b' \neq \perp \wedge tr \notin Faulty(Auth, b') \\
& \qquad \wedge BB_{rec}([b]) \in tr \wedge BB_{woS}([\tilde{b}]) \in tr \wedge BB_{pkD}([pk_D]) \in tr \\
& \qquad \wedge BB_H([H]) \in tr \\
\xrightarrow{(2)} & \exists[pk_D^u].\ BB_{rec}([b]) \in tr \wedge BB_{woS}([\tilde{b}]) \in tr \\
& \qquad \wedge BB_{pkD}([pk_D]) \in tr \wedge BB_H([H]) \in tr \wedge [\tilde{b}] = ver_L([b], [pk_D^u]) \\
& \qquad \wedge [pk_D^u] \subseteq_l [pk_D] \\
\xrightarrow{(3)} & \exists[sk_D].\ BB_{rec}([b]) \in tr \wedge BB_{woS}([\tilde{b}]) \in tr \\
& \qquad \wedge BB_{pkD}([pk_D]) \in tr \wedge BB_H([H]) \in tr \wedge [b] = [\{\tilde{b}\}_{sk_D}] \\
& \qquad \wedge [pk(sk_D)] = [pk_D^u] \wedge [pk_D^u] \subseteq_l [pk_D].
\end{aligned}$$

Steps (1) holds as the bulletin board is honest by topology T_{MV}, thus when the recorded ballots are published, then the lists $[\tilde{b}]$, $[pk_D]$, and $[H]$ have previously been published. As $tr \notin Faulty(Auth, b')$ for some ballot $b' \neq \perp$, both disjunct specified in the definition of $Faulty(Auth, b')$ for the case $b' \neq \perp$ must be false in tr. In particular, as the second disjunct is false, all recorded ballots must contain a unique valid signature, i.e., corresponding to a public key that is contained in $[pk_D]$, indicated by Step (2). Step (3) holds by the definitions of ver_L.

Let b be an arbitrary recorded ballot. By the above formula, it has the form $b = \{\tilde{b}\}_{sk_D}$ for some signing key sk_D such that $pk(sk_D) \in [pk_D]$ (thus $ver(b, pk(sk_D)) \neq \perp$) and such that no other recorded ballot is associated with the same key sk_D. As the bulletin board *BB* is honest by topology T_{MV} and as there are no parallel sessions, *BB* only publishes the list $[pk_D]$ from its initial knowledge in the signal BB_{pkD}. By the shared initial knowledge and the setup of the protocol (there are no parallel sessions), for each verification key pk_D in the list $[pk_D]$ from *BB*'s initial knowledge, there is exactly one device which knows the corresponding signing key sk_D and one voter H who is associated with this device. Moreover, the device and voter associated with each verification key are distinct. Also, the verification key pk_D is together with the voter H in the list in $Corr([H, pk_D])$, and H is in the list $[H]$. Furthermore, these list are correctly published on the bulletin board *BB* as *BB* is honest by topology T_{MV}. From this, all conjuncts in the property's definition (Definition 9) except for the sends already follow.

We next establish that each sk_D is only known to one device D at all times during the execution. First, by the specified initial knowledge, the signing key is only known to this device at the protocol's start. Second, by the topology T_{MV} the agent instantiating the device is honest and follows its specification. Therefore, D only sends one message, which contains the term sk_D as a signature. By our equational theory, it is not possible to extract a signing key from a signed message. Therefore, no agent other than D knows sk_D during the protocol execution.

We have established that $b = \{\tilde{b}\}_{sk_D}$ is recorded on the bulletin board and that no agent except D knows the term sk_D. Thus, D must have computed b. As we have just argued that D is honest, D only sends one message to its associated voter H on a secure channel. Therefore, in all traces where b is recorded on the bulletin board, the unique voter H must have forwarded b, that is H must have sent a message containing b to some agent A. Thus, $send(H, A, m)$ is in the trace with $m \vdash b$. The above reasoning holds for any ballot. Furthermore, we argued that for each recorded ballot, there is a unique associated signing key and device, i.e., distinct from those associated with the other recorded ballots, and that each device has a unique associated voter. Therefore, it follows that for any two distinct recorded ballots the required signal *send* is recorded with a distinct voter. □

Lemma 15. *MixVote* satisfies *EligVerif* when run in the topologies $T_{MV}^{Auth^+H^+}$ and $T_{MV}^{H^+}$.

Proof. Let tr be a trace in $TR(Pr_{MV}, T_{MV}^{Auth^+H^+}) \cup TR(Pr_{MV}, T_{MV}^{H^+})$ such that $q_1 = verify_{A1}(A, \exists [pk_D^u].[pk_D^u] \subseteq_l [pk_D] \wedge [\tilde{b}] = ver_L([b], [pk_D^u]), true) \in tr$ and $q_2 = verify_{A2}(A, ver_{zk}(p, [\tilde{b}], [v], pk_{Auth}), true) \in tr$ and let i and j be two natural numbers in $\{1, \ldots, n\}$, where n is the length of the lists $[v]$ and $[b]$.

$$\begin{aligned}
& q_1 \wedge q_2 \wedge i, j \in \{1, \ldots, n\} \\
\overset{(1)}{\Longrightarrow} & \exists [sk_D], [r], \pi.\ q_1 \wedge q_2 \wedge i, j \in \{1, \ldots, n\} \\
& \wedge [b] = \pi[\{\{v\}^r_{pk_{Auth}}\}_{sk_D}] \wedge [pk(sk_D)] = [pk_D^u] \wedge [pk_D^u] \subseteq_l [pk_D] \\
& \qquad \wedge BB_{rec}([b]) \in tr \wedge BB_{tal}([v]) \in tr \wedge BB_{pkD}([pk_D]) \in tr \\
\overset{(2)}{\Longrightarrow} & \exists [H].\ q_1 \wedge q_2 \wedge i, j \in \{1, \ldots, n\} \wedge [b] = \pi[\{\{v\}^r_{pk_{Auth}}\}_{sk_D}] \\
& \qquad \wedge [pk(sk_D)] = [pk_D^u] \wedge [pk_D^u] \subseteq_l [pk_D] \wedge BB_{rec}([b]) \in tr \\
& \qquad \wedge BB_{tal}([v]) \in tr \wedge BB_{pkD}([pk_D]) \in tr \wedge BB_{woS}([\tilde{b}]) \in tr \\
& \qquad \wedge BB_H([H]) \in tr \\
\overset{(3)}{\Longrightarrow} & \exists i', j', P_1, P_2, m_1, m_2.\ i, j \in \{1, \ldots, n\} \\
& \wedge [b] = \pi[\{\{v\}^r_{pk_{Auth}}\}_{sk_D}] \wedge [pk(sk_D)] = [pk_D^u] \wedge [pk_D^u] \subseteq_l [pk_D] \\
& \qquad \wedge BB_{rec}([b]) \in tr \wedge BB_{tal}([v]) \in tr \wedge BB_{pkD}([pk_D]) \in tr \\
& \qquad \wedge BB_{woS}([\tilde{b}]) \in tr \wedge BB_H([H]) \in tr \\
& \qquad \wedge send([H]_{i'}, P_1, m_1) \in tr \wedge send([H]_{j'}, P_2, m_2) \in tr \\
& \qquad \wedge m_1 \vdash [b]_i \wedge m_2 \vdash [b]_j \wedge (i \neq j \implies [H]_{i'} \neq [H]_{j'}).
\end{aligned}$$

Step (1) holds by the same reasoning as in the proof of Lemma 13. Next, Step (2) holds as *BB* is honest by the topology T_{MV}, thus when the signals BB_{tal} and BB_{rec} are recorded in the trace, then the signals BB_{woS} and BB_H have also been recorded. Step (3) holds by the reasoning in the proof of Lemma 14, where we derived from Step (3) that *Uniqueness*(*Auth*) holds. As for each vote v in $[v]$ there is a ballot b in $[b]$ such that $b = \{\{v\}^r_{pk_{Auth}}\}_{sk_D}$, it follows that each such *send*-signal in Step (3) contains a unique vote from the final tally.

As we have argued in the proof of Lemma 14 that only each voter's device can compute such a ballot with the valid verification key, for each voter H to send a message containing the ballot, H must have previously learned it from his device D. Moreover, as each device D is honest by the topology T_{MV}, D only computes the ballot $b = \{\{v\}^r_{pk_{Auth}}\}_{sk_D}$ when it has previously received v from its associated voter H on a secure channel. Thus, for each voter H to send a ballot of the form $b = \{\{v\}^r_{pk_{Auth}}\}_{sk_D}$, H must have previously sent the corresponding vote v to his device. In particular, as there are no parallel sessions and D only receives and sends one message, no message confusion is possible. We conclude that each (distinct) vote in the final tally must have been sent by a (distinct) voter to his device. □

Lemma 16. *MixVote* satisfies *VoterA*(*Auth*) when run in the topologies $T_{MV}^{Auth^+H^+}$ and $T_{MV}^{H^+}$.

Proof. Let tr be a trace in $TR(Pr_{MV}, T_{MV}^{Auth^+H^+}) \cup TR(Pr_{MV}, T_{MV}^{H^+})$. As all traces in $TR(Pr_{MV}, T_{MV}^{Auth^+H^+}) \cup TR(Pr_{MV}, T_{MV}^{H^+})$ satisfy *Uniqueness*(*Auth*) by Lemma 14 and as the preconditions of Theorem 2 hold (i.e., the protocol does not allow re-voting and a voter who abstains does not send any messages), we use Theorem 2 to infer that tr satisfies *VoterA*(*Auth*). □