SoK: A Unified Framework for Analyzing Coercion Resistance in Electronic Voting

Wonseok Choi
DGIST
Daegu, Republic of Korea
wonseok@dgist.ac.kr

Won Kim
Korea University
Seoul, Republic of Korea
munsanwon2@korea.ac.kr

Changmin Lee
Korea University
Seoul, Republic of Korea
changminlee@korea.ac.kr

***Abstract*—Electronic voting (E-voting) can improve the accessibility and efficiency of elections, but its reliance on electronic systems also introduces security and privacy threats. In remote e-voting, adversaries may observe voters' behavior and obtain some electronic information, making voters more vulnerable to coercion. Hence, coercion resistance is a security property of e-voting protocols which ensures that a voter can deviate from an adversary's instructions. However, existing e-voting protocols establish coercion resistance under different assumptions about the protocol settings, such as election environments, trust assumptions, and adversarial capabilities. In particular, e-voting protocols involve diverse procedures and interactions, resulting in different protocol syntaxes. Consequently, it is difficult to determine whether their coercion resistance guarantees hold in practical election settings and which protocols provide stronger protection against coercion.**

In this work, we present a unified framework for systematically analyzing coercion resistance across different e-voting protocols. We introduce a unified syntax for e-voting protocols that captures diverse coercion-resistant protocols. This syntax provides a common execution model in which adversarial capabilities can be specified consistently across different protocols. Using our syntax, we systematically express existing protocols within this framework. We also adapt standard definitions of verifiability and ballot privacy of e-voting protocols to our syntax. Using these definitions, we compare the verifiability and ballot privacy of existing coercion-resistant e-voting protocols.

We then extend existing definitions of coercion resistance to our unified e-voting syntax. We systematically classify coercion-resistance settings along five axes: corruption bounds, corruption modes, coercion phases, timing restrictions, and the number of coercion interactions. We establish implication relations among the resulting coercion resistance notions to characterize their relative strength. Using this classification, we evaluate existing e-voting protocols by proving the maximality of coercer-capability settings under which each protocol achieves coercion resistance. Our framework thereby enables direct and fine-grained comparison of coercion-resistant e-voting protocols under our common model.

1. Introduction

Electronic voting (e-voting) has attracted considerable attention for its potential to improve the accessibility and efficiency of elections, particularly by enabling voters to participate remotely. However, unlike traditional voting conducted in supervised polling stations, remote e-voting may take place in uncontrolled environments where adversaries can observe or influence voters' behavior. Consequently, coercion resistance has been raised as a crucial security requirement for ensuring voters' freedom of choice [1]. Informally, coercion resistance ensures that voters can deviate from a coercer's instructions, such as voting for a particular candidate or abstaining from voting, without the coercer being able to reliably determine whether those instructions were followed. Given its importance, coercion resistance has been extensively studied, leading to numerous e-voting protocols and formal security definitions [2], [3].

However, existing e-voting protocols have often been designed and analyzed under security models tailored to their specific structures, resulting in different protocol syntaxes and assumptions about election environments and adversarial capabilities. In particular, these models may exclude realistic coercion scenarios, leaving room for real-world coercion despite of their formal proof of coercion resistance under their assumptions. Moreover, the lack of a common framework makes it difficult to compare the security guarantees of different protocols, even when they are all claimed to achieve coercion resistance. Consequently, it remains unclear which guarantees hold under practical election settings and which protocols provide stronger protection against coercion.

For example, last-minute voter coercion have been recently highlighted in the literature, in which a coercer forces a voter to cast a particular ballot shortly before the voting period ends, leaving insufficient time to recover through re-voting [4]. This threat is particularly relevant to re-voting-based systems, such as IVXV used in Estonian elections, which rely on voters being able to replace coerced ballots with their intended choices. Such a realistic attack scenario illustrates how existing coercion-resistance guarantees may fail to capture adversarial behavior that falls outside their assumed threat models, highlighting the need for a more comprehensive analysis of coercer capabilities.

Nevertheless, last-minute coercion represents only one of many realistic coercion scenarios, and addressing this particular attack does not guarantee complete protection against every coercion. Although previous studies have proposed general definitions and threat models for coercion resistance, they have not systematically characterized how different coercer capabilities affect the security guarantees of e-voting protocols across various election settings [5], [6]. This highlights the need for a unified framework that comprehensively classifies different coercion scenarios and enables systematic comparisons of coercion resistance across different protocols.

1.1. Our contribution

Our main contributions are summarized in Table 1.

Supplementary proofs of our contributions are also available at:

https://github.com/munsanwon2/e-voting_unified_coercion_resistance_proof

- **Unified E-voting Syntax.** We introduce a unified syntax for e-voting protocols consisting of 11 algorithms. Our syntax accommodates different protocol structures, while preserving protocol-specific interactions and observable information. To the best of our knowledge, we express all existing e-voting protocols which claimed to achieve coercion resistance within this syntax, establishing a common execution model for their security analysis.
- **Unified Analysis of Verifiability and Ballot Privacy.** We adapt standard definitions of weak and strong verifiability and ballot privacy to our unified syntax, accounting for different voting procedures and election rules. We systematically analyze the considered protocols under these common definitions. In particular, we provide formal security analyses for protocols whose original works either lacked corresponding proofs or employed different security notions. This enables a direct comparison of their verifiability and ballot-privacy guarantees under explicitly stated assumptions.
- **Unified Definition and Classification of Coercion Resistance.** We extend the existing game-based definition of coercion resistance to our unified e-voting syntax and classify coercer capabilities along five axes: corruption bounds, corruption modes, coercion phases, timing restrictions, and the number of coercion interactions. By incorporating these capabilities into a unified coercion resistance definition, we capture coercion resistance under different adversarial settings within a common framework. We further establish implication relations among the different coercer capabilities, enabling systematic comparisons of their relative strength.
- **Coercion-Resistance Analysis and Maximality.** We systematically analyze existing e-voting protocols claimed to achieve coercion resistance under our unified definition. For each protocol, we prove coercion resistance under the classified coercer capabilities (or prove that the protocol does not satisfy the coercion resistance) and establish their maximality along each axis through corresponding impossibility results. These results characterize the maximal coercer capabilities against which each protocol achieves coercion resistance, enabling direct and fine-grained comparisons of their security guarantees.

1.2. Related work

Existing work has pursued unification through general security definitions, hierarchies of notions, and comparisons using a common election syntax. A remaining obstacle is to preserve the interactions of structurally different voting protocols within a common computational model, while making coercer capabilities comparable throughout the election.

Computational definitions. Juels, Catalano, and Jakobsson [1] formalized coercion resistance against attacks including forced abstention, randomization, and credential disclosure. Their experiment incorporates an evasion mechanism based on fake credentials, which ties its formulation to a particular class of voting systems. Küsters, Truderung, and Vogt [3] gave a more general computational definition: a voter must have a counterstrategy that achieves its goal while remaining difficult to distinguish from obedience. Their quantitative formulation accounts for information revealed by the election outcome and already permits interactive coercion and dishonest participants. However, the protocol programs, communication interfaces, and scheduling assumptions are supplied separately for each election system. This abstraction supports broad applicability, but leaves election interfaces and coercion opportunities to be aligned when comparing structurally different protocols. We retain KTV's security objective and make these choices explicit within a common operational syntax.

Symbolic frameworks. Delaune, Kremer, and Ryan [49] formalized vote privacy, receipt-freeness, and coercion resistance using observational equivalence in the applied π-calculus, and established implications between them. Their coercion-resistance definition excludes randomization and forced-abstention attacks, illustrating that the same security label can cover different attack classes. Küsters and Truderung [2] developed a model-independent epistemic framework supporting different voter goals, coercion strategies, and environmental assumptions. Heather and Schneider [5] parameterized coercion resistance by instruction sets and observable events in the process algebra CSP, and proved a hierarchy that includes randomization and forced abstention. Dreier, Lafourcade, and Lakhnech [50] proposed a modular privacy taxonomy covering interaction with the coercer, insider attacks, forced abstention, and counterbalancing behavior. These works already establish the value of families of security notions. Their symbolic

TABLE 1. E-VOTING PROTOCOLS CONSIDERED IN OUR ANALYSIS. IN CORRUPTION SETS, R, T, BB, V, O, D, AND E DENOTE REGISTRARS, (THRESHOLD-COMPROMISED) TALLIERS, BULLETIN-BOARD OWNERS, NON-TARGET VOTERS, PASSIVE OBSERVERS, NON-TARGET DEVICE OWNERS, AND PURE EXPERTS, RESPECTIVELY.

Protocol	Paper	Verifiability	Ballot privacy	Coercion resistance
JCJ-based protocols				
JCJ protocol	[1]	Weak	Honest-majority talliers, anonymous channel	$\Omega_{\{V,O,T\},\mathsf{adapt},\{\mathsf{I}\},\mathsf{full},L}$
Civitas	[7]	No	Honest reg./talliers, anonymous casting	$\Omega_{\varnothing,\mathsf{adapt},\{\mathsf{R},\mathsf{I}\},\mathsf{full},L}$
AFT2010	[8]	No	Honest issuer/mix, anonymous channel	$\Omega_{\{V,O\},\mathsf{adapt},\{\mathsf{R},\mathsf{V},\mathsf{I}\},\mathsf{full},0}$
CANS2010/ABRTY	[9]	No	Honest registration/mix, anonymous channel	$\Omega_{\{V,O\},\mathsf{adapt},\{\mathsf{R},\mathsf{V},\mathsf{I}\},\mathsf{full},0}$
Schlaepfer et al. (SHKS)	[10]	Weak	Honest registration/talliers, anonymous casting	No
Cobra	[11]	No	Private registration, honest authorization/talliers	$\Omega_{\{V,O\},\mathsf{adapt},\{\mathsf{R},\mathsf{V},\mathsf{I}\},\mathsf{full},0}$
CHide	[12]	Weak	DDH/ROM; native joint ballot proofs and secure DKG/mixnets	$\Omega_{\{V\},\mathsf{adapt},\{\mathsf{I}\},\mathsf{full},L}$
Aranha–Battagliola–Roy (ABR)	[13]	No	DDH/ROM; native joint proofs, CGates and mixnets	$\Omega_{\{V\},\mathsf{adapt},\{\mathsf{I}\},\mathsf{full},L}$
Efficient Cleansing	[14]	Weak	Native NM-CPA ballot encoding, Fiat–Shamir proofs, private registration	$\Omega_{\{V,O\},\mathsf{adapt},\{\mathsf{R},\mathsf{V},\mathsf{I}\},\mathsf{full},0}$
PACBS-JCJ protocol	[15]	No	Anonymous channel	No
zkVoting	[16]	Strong	Trusted registrar, IND-CPA/ZK	No
LRS-JCJ protocol	[17]	Strong	NM-CPA, NIZK	$\Omega_{\{V,O\},\mathsf{adapt},\{\mathsf{R},\mathsf{V},\mathsf{I}\},\mathsf{full},0}$
Votegral	[18], [19]	Weak	≥ 1 honest mixer, trusted client	$\Omega_{\{V,O\},\mathsf{adapt},\{\mathsf{R},\mathsf{V},\mathsf{I}\},\mathsf{full},0}$ $\Omega_{\{V,O\},\mathsf{adapt},\{\mathsf{I}\},\mathsf{full},L}$
SKHS	[20]	Weak	JCJ registration/mixing/tally trust	$\Omega_{\{V,O\},\mathsf{adapt},\{\mathsf{I}\},\mathsf{full},L}$
Araujo–Traore (AT13)	[21]	No	Honest registration/tally, anonymous channel	$\Omega_{\{V,O\},\mathsf{adapt},\{\mathsf{R},\mathsf{V},\mathsf{I}\},\mathsf{full},0}$
Araujo et al. (MAC)	[22]	No	Honest MAC issuance/talliers; anonymous voting	$\Omega_{\{V,O\},\mathsf{adapt},\{\mathsf{R},\mathsf{V},\mathsf{I}\},\mathsf{full},0}$
Eos	[23]	No	Anonymous signatures; honest mixer/talliers	$\Omega_{\{V,O,D\},\mathsf{adapt},\{\mathsf{R},\mathsf{V},\mathsf{I}\},\mathsf{full},0}$
Kulyk et al. proxy voting	[24]	No	Honest registrar/mixer, unlinkable delegations	No
FHE-JCJ	[25]	No	Threshold FHE secrecy, honest mix; ZK	No
JCJ–Selene	[26]	Weak	JCJ trust, private tracker delivery; honest mix	$\Omega_{\{V,O\},\mathsf{adapt},\{\mathsf{I}\},\mathsf{full},L}$
Re-voting protocols				
AKLM	[27]	Weak	No; independent-contribution copying	$\Omega_{\{V,O\},\mathsf{adapt},\{\mathsf{R},\mathsf{V},\mathsf{I}\},\mathsf{full},0}$
LHK	[28]	Weak	Anonymous channel, trusted client	$\Omega_{\varnothing,\mathsf{adapt},\{\mathsf{R},\mathsf{V},\mathsf{I}\},\mathsf{full},0}$
IVXV	[29], [30]	Weak	t-out-of-n talliers, anonymization	$\Omega_{\{V,O,T\},\mathsf{adapt},\{\mathsf{R},\mathsf{V},\mathsf{I}\},\mathsf{full},0}$
VoteAgain	[31]	Weak	Native VoteAgain; DDH/ROM	$\Omega_{\{V,O\},\mathsf{adapt},\{\mathsf{R},\mathsf{V},\mathsf{I}\},\mathsf{full},0}$
Loki	[4]	Strong	Honest registrar/server/BB/tallier	$\Omega_{\{V,O\},\mathsf{adapt},\{\mathsf{R},\mathsf{V},\mathsf{I}\},\mathsf{full},0}$
CR-Loki	[32]	Strong	IND-RCCA; source UC-secure NIZK realization	No
DeVoS	[33]	Weak	Native authenticated DeVoS; source mixnet realization	$\Omega_{\{V,O\},\mathsf{adapt},\{\mathsf{I}\},\mathsf{full},L}$
FiltrumVote	[34]	Weak	Honest BB/device, t-out-of-n talliers	$\Omega_{\{O\},\mathsf{adapt},\{\mathsf{R},\mathsf{V},\mathsf{I}\},\mathsf{full},0}$
Selections	[35]	No	Anonymous channel, t-out-of-n talliers	$\Omega_{\{V,O\},\mathsf{adapt},\{\mathsf{R},\mathsf{V},\mathsf{I}\},\mathsf{full},0}$
Revisited VoteAgain	[36]	Weak	Native revised VoteAgain; original vote proofs and tally primitives	No
Amun	[37]	No	Source: at most one of three authorities corrupted; DDH/ROM	$\Omega_{\{V,O\},\mathsf{adapt},\{\mathsf{R},\mathsf{V},\mathsf{I}\},\mathsf{full},0}$
Farzaliyev–Willemson	[38]	Weak	Honest private BB; IND-CPA/ZK and audit	$\Omega_{\{V,O\},\mathsf{adapt},\{\mathsf{R},\mathsf{V},\mathsf{I}\},\mathsf{full},0}$
Blockchain-based protocols				
Dimitriou protocol	[39]	Weak	No; public key release before terminal tally	No
Spadafora–Longo–Sala protocol	[40]	Weak	≥ 1 of 2 honest authorities, DDH	$\Omega_{\{V,O\},\mathsf{adapt},\{\mathsf{I}\},\mathsf{full},L}$
Sarier BBEV	[41]	Weak	Anonymous mixing	$\Omega_{\varnothing,\mathsf{adapt},\{\mathsf{I}\},\mathsf{full},L}$
Yin et al. scheme	[42]	Strong	No; weighted-only result with public choice openings	$\Omega_{\{E,O\},\mathsf{adapt},\{\mathsf{R},\mathsf{V},\mathsf{I}\},\mathsf{full},0}$ $\Omega_{\{E,O,T\},\mathsf{adapt},\{\mathsf{R},\mathsf{V},\mathsf{I}\},\mathsf{full},0}$
ZKP-E-Vote	[43]	No	t-out-of-n talliers, ZK-PoK	No
CAI-based protocols				
Finogina–Herranz (S_1, S_2)	[44]	No	Compatible BPRIV host; source-private S_1, S_2 casting	$\Omega_{\{V,O\},\mathsf{adapt},\{\mathsf{I}\},\mathsf{full},L}$
Finogina–Herranz limited-voter scheme	[45]	No	Honest registrar/BB/tallier/OED, private channel, DDH/ROM, NIZK	$\Omega_{\{V,O\},\mathsf{adapt},\{\mathsf{I}\},\mathsf{full},L}$
Kho–Heng–Tan–Chin scheme	[46]	Weak	IND-CCA; private channels; owner/session-bound admission; simulatable tally NIZK	No
Other constructions				
Anamorphic E-Voting	[47]	No	Source PKE/AME/ZK assumptions; protected channels	$\Omega_{\{O\},\mathsf{adapt},\{\mathsf{I}\},\mathsf{full},L_D}$
Backes–Gagne–Skoruppa building blocks	[48]	No	Depends on host; independent mobile channel	$\Omega_{\{V,O,T\},\mathsf{adapt},\{\mathsf{I}\},\mathsf{full},L}$

semantics, however, do not quantify the probabilities that determine a computational coercer's distinguishing advantage. Our framework combines a computational guarantee with explicit restrictions on corruption and contact throughout the election.

Composable incoercibility. Unruh and Müller-Quade [51] introduced a general framework for incoercible reactive protocols with a universal composition theorem. Alwen et al. [52] developed universally composable (UC) incoercibility with local deception strategies and several coercion types, including receipt-freeness and active coercion. These frameworks support general interaction and provide compositional guarantees relative to an ideal functionality. Applying them to an election still requires specifying the functionality, communication interfaces, and permitted leakage. Their generality does not by itself supply a common election syntax for comparing how different registration, voting, and verification procedures expose voters to the same coercer capabilities.

Comparisons of definitions. Haines and Smyth [53] compared four computational definitions within a common election syntax and proposed repaired or simplified formula-

tions. The scope of this unification is limited: the authors explicitly acknowledge that their syntax cannot model every voting system. Registration produces credentials in a single registrar-side call, while voting is represented by local ballot construction. These interfaces do not explicitly represent voter–registrar exchanges or voter–authority interaction during ballot construction. Their sequential presentation of KTV assumes that individual ballots cannot be observed as they are cast, and verification interactions are excluded. Consequently, extending this comparison to interactive protocols requires preserving their intermediate messages, local states, and opportunities for adversarial intervention. Simply representing an interactive procedure by its final output can omit precisely the observations on which a coercion attack depends. The resulting comparison is therefore limited to the protocol behavior captured by the chosen syntax and falls short of a unified treatment of diverse election procedures.

Recent refinements. Cortier, Gaudry, and Yang [12] showed that the cleansing phase of JCJ can leak additional information and revised the definition to account for revoting and authority-generated dummy ballots. Their experiment captures richer voting histories, but retains a credential-based evasion strategy and a prescribed ordering of registration, corruption, coercion, and ballot processing. Rønne, Finogina, and Herranz [54] introduced definitions with oracles for tools such as timed commitments and trusted hardware, together with relations between variants of these definitions. Their analysis concerns vote casting and explicitly excludes forced abstention and randomization. These refinements address particular protocol mechanisms or stages, but do not supply a common execution model covering interactions throughout the election and the associated corruption and contact restrictions.

Threat assessment. Longo et al. [6] adapted STRIDE, LINDDUN, and OWASP risk ratings to assess threats to e-voting systems. Their work connects security requirements to attacker capabilities and practical risks. Such assessments help identify relevant threats, but a risk rating does not itself establish that a voter has a successful counterstrategy or that resistance is computationally indistinguishable from obedience. Our experiment provides a formal setting in which those obligations can be stated for the identified capabilities.

Positioning of our work. These approaches leave a gap between general security notions and a unified computational treatment of the interactions through which different voting protocols expose voters to coercion. Our syntax explicitly represents registration exchanges, voting authorization, ballot validation, and verification, while preserving local states and observable messages throughout the election. The execution permits repeated and interleaved sessions according to the protocol's policies, so coercion can be analyzed at the relevant steps of these procedures. On this basis, our framework organizes five operational choices within one computational experiment for a target voter: which parties may be corrupted, whether corruption is static or adaptive, which election phases permit coercion, when contact is permitted within those phases, and how many direct exchanges are allowed. Ending direct contact does not erase information already obtained or terminate an instruction that continues to execute locally. With the election environment and public-information rules fixed, the framework makes positive counterstrategies and attacks against stronger capabilities comparable across protocols. The resulting classification identifies the assumptions supporting a coercion-resistance claim and the additional capabilities under which that claim fails.

1.3. Technical Overview

To systematically compare coercion resistance across e-voting protocols with different procedures and adversarial assumptions, we develop a unified framework in three steps.

Unified E-voting Syntax. We represent the main operations of different e-voting protocols by 11 algorithms with common conventions for messages, local states, and bulletin-board access. Their roles are as follows.

- Setup initializes the election parameters and the private states of election authorities and the bulletin board.
- Apply performs the voter's registration request.
- Register processes registration requests from the voter and updates the registrar's state.
- Request prepares a ballot-authorization request for voting and the voter's private continuation state.
- Authorize processes the voter's voting request according to eligibility, usage, and revoting rules.
- Vote constructs a ballot for the voter's selected choice.
- VerifyVote checks whether a ballot is posted on the bulletin board.
- Valid processes ballot admission and other prescribed validation events, updating the internal ballot records.
- Publish exposes the public view of the bulletin board from its internal records.
- Tally computes the election result and its proof from the final records.
- VerifyTally publicly verifies the announced result from its proof.

Extended Coercion-Resistance Definition. We define coercion resistance by retaining the central idea of Küsters, Truderung, and Vogt [3, KTV]: a voter should be able to achieve its own voting goal without enabling the coercer to distinguish resistance from obedience. In our experiment, a challenger samples a bit $b \in \{0, 1\}$. The target voter follows the honest protocol until the first coercion; thereafter, it obeys the coercer's instructions if $b = 0$, or follows a counterstrategy to avoid coercion if $b = 1$. The coercer observes the information available to it and guesses b. Coercion resistance requires that a ballot for the voter's intended choice be counted in the resistant world with overwhelming probability and that the coercer's distinguishing advantage

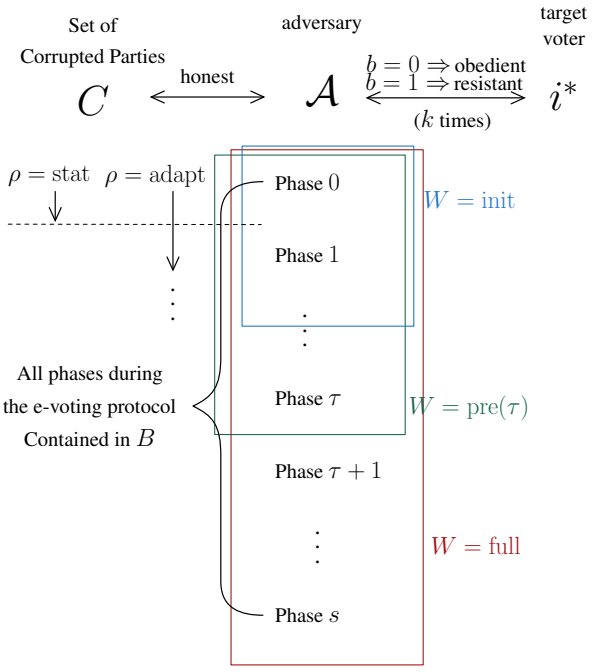


Figure 1. Classified coercer abilities.

be at most $\delta/2$ up to a negligible term, where $\delta \in [0, 1]$ is the tolerated distinguishability between obedience and resistance. The difference from KTV is not this security objective, but its instantiation: we define the experiment over the unified protocol syntax and explicitly specify classified coercer abilities.

Classified Coercer Abilities. To compare protocols under explicitly stated adversarial settings, we classify the coercer's capabilities along five axes, illustrated in Figure 1.

- **Corruption bound** C specifies which non-target parties, such as registrars or talliers, may be corrupted.
- **Corruption mode** ρ distinguishes static corruption before setup from adaptive corruption during execution.
- **Coercion phase** B specifies whether direct contact is permitted during registration, voting, individual verification, or their combinations.
- **Time model** W restricts contact to the first k eligible times, to eligible times up to a cutoff τ, or to any eligible time.
- **Number of coercions** k bounds the total number of direct exchanges between the coercer and the target voter.

2. Unified definitions of e-voting protocols and their basic properties

In this section, we first define e-voting protocols by a unified syntax with 11 algorithms. Then, we define verifiability and ballot privacy of e-voting protocols by extending the previous standard definitions into our unified syntax.

2.1. Electronic voting

Definition 1 (E-voting protocol)**.** Let λ be a security parameter. An election descriptor E specifies the eligible voter set $\mathcal{U}$, the vote space $\mathcal{M}$, and any other fixed public election parameters, including the registration, authorization, authorization-scheduling, counting, revoting, validation-event, publication, bulletin-board access, and individual-verification policies.

An e-voting protocol is a tuple of PPT algorithms

$$\Pi_{\mathsf{voting}} = \big(\mathsf{Setup}, \mathsf{Apply}, \mathsf{Register}, \mathsf{Request}, \mathsf{Authorize}, \mathsf{Vote}, \mathsf{VerifyVote}, \mathsf{Valid}, \mathsf{Publish}, \mathsf{Tally}, \mathsf{VerifyTally}\big)$$

with the following syntax:

$$\begin{aligned}
\mathsf{Setup}(1^\lambda, \mathsf{E}) &\to (\mathsf{pp}, \mathsf{td}, \mathsf{rd}, \mathsf{BBd}),\\
\mathsf{Apply}(\mathsf{pp}, \mathsf{rst}, i, \omega_{\mathsf{R}}, \mathsf{ans}_{\mathsf{R}}; r_{\mathsf{QR}}) &\to (\mathsf{req}_{\mathsf{R}}, \omega'_{\mathsf{R}}, z_{\mathsf{R}}),\\
\mathsf{Register}(\mathsf{pp}, \mathsf{rd}, i, \mathsf{req}_{\mathsf{R}}; r_{\mathsf{R}}) &\to (\mathsf{ans}_{\mathsf{R}}, \mathsf{rd}', \mathsf{rst}'),\\
\mathsf{Request}(\mathsf{pp}, \mathsf{rst}, i, \mathsf{cred}_i, m; r_{\mathsf{Q}}) &\to (\mathsf{req}, \omega),\\
\mathsf{Authorize}(\mathsf{pp}, \mathsf{rd}, \mathsf{rst}, i, \mathsf{req}; r_{\mathsf{A}}) &\to (\mathsf{ans}, \mathsf{rd}', \mathsf{rst}'),\\
\mathsf{Vote}(\mathsf{pp}, \mathsf{cred}_i, m, \omega, \mathsf{ans}; r_{\mathsf{V}}) &\to \beta,\\
\mathsf{VerifyVote}(\mathsf{pp}, \mathsf{cred}_i, m, r_{\mathsf{V}}, \beta, \mathsf{PBB}) &\to b_{\mathsf{vote}} \in \{0, 1\},\\
\mathsf{Valid}(\mathsf{pp}, \mathsf{BBd}, \mathsf{rst}, \mathsf{IBB}, x) &\to (b_{\mathsf{val}}, \mathsf{BBd}', \mathsf{IBB}'),\\
\mathsf{Publish}(\mathsf{pp}, \mathsf{rst}, \mathsf{IBB}) &\to \mathsf{PBB},\\
\mathsf{Tally}(\mathsf{pp}, \mathsf{td}, \mathsf{rst}, \mathsf{IBB}) &\to (R, \pi),\\
\mathsf{VerifyTally}(\mathsf{pp}, \mathsf{PBB}, R, \pi) &\to b_{\mathsf{tal}} \in \{0, 1\}.
\end{aligned}$$

Here pp denotes the public election parameters, td denotes the private state held by the tallier(s), rd denotes the private mutable state held by the registration and authorization authorities, and rst denotes their evolving public state. The state

$$\mathsf{BBd} \in \mathcal{S}_{\mathsf{VS}}$$

denotes private mutable state held by the ballot-validation or voting server.

In various e-voting protocols, these 11 algorithms are specified to representation the phases of election as follows.

Stateful bulletin board. The public parameters determine an authenticated stateful bulletin-board functionality $\mathcal{F}_{\mathsf{BB}}[\mathsf{E}]$ whose private state is of the form

$$\mathsf{st}_{\mathsf{BB}} = \big(\mathbf{B}, \mu_{\mathsf{bal}}, \mu_{\mathsf{ver}}\big).$$

Here $\mathbf{B} \in \mathcal{A}^*_{\mathsf{rec}}$ is the append-only sequence of internal ballot records, μ_{bal} is protocol-prescribed ballot-processing metadata, and μ_{ver} is private individual-verification metadata, such as session status, access counters, admissible access times, and latest-ballot indicators.

The internal capability IBB enables the protocol-prescribed validation and tallying operations, while PBB enables only the public, audit, and selective-read interfaces prescribed by E. We write

$$\mathsf{Records}(\mathsf{IBB}) := \mathbf{B}.$$

In particular, a public capability may provide interfaces of the form

$$\mathsf{ReadPub}(\mathsf{PBB}) \to \mathsf{view}_{\mathsf{pub}},$$
$$\mathsf{ReadVer}(\mathsf{PBB}, \xi) \to y \in \mathcal{Y}_{\mathsf{E}} \cup \{\bot\},$$
$$\mathsf{ReadAudit}(\mathsf{PBB}) \to \mathsf{view}_{\mathsf{aud}},$$

where ξ is a protocol-specific verification index or capability. A call to ReadVer may atomically update μ_{ver}, but it must not modify $\mathbf{B}$, the canonical public view, or any value used to determine the election result.

Each top-level algorithm is executed by one logical party as a PPT oracle algorithm. A call to an interface of $\mathcal{F}_{\mathsf{BB}}[\mathsf{E}]$ is a black-box subroutine call and does not introduce an additional component of Π_{voting}. The calling party learns only the interface output and never obtains the private bulletin-board state.

For a protocol requiring only a static, globally readable board, one may set

$$\mu_{\mathsf{ver}} = \bot, \qquad \mathsf{PBB} = (\mathsf{rst}, \mathbf{B}),$$

and interpret every bulletin-board read as ordinary deterministic access to this string.

Registration. During registration, an eligible voter i interacts with the registrar to obtain a credential cred_i required for subsequent voting. A registration session consists of a finite sequence of alternating voter-side Apply and registrar-side Register invocations.

Initially, the voter has no pending registration state or registrar response, so

$$\omega_{\mathsf{R}} = \bot, \qquad \mathsf{ans}_{\mathsf{R}} = \bot.$$

In each round, the voter invokes Apply to generate a registration request $\mathsf{req}_{\mathsf{R}}$ to be sent to the registrar. The request may contain, for example, an eligibility claim, authentication information, or a response to a previous challenge. The voter also maintains a private continuation state ω_{R}, which stores information needed in subsequent rounds.

Upon receiving $\mathsf{req}_{\mathsf{R}}$, the registrar invokes Register to process the request and returns a response $\mathsf{ans}_{\mathsf{R}}$. The registrar maintains its private registration state in rd and updates the public registration records in rst. The voter then invokes Apply again using the received response and its updated private state. This interaction continues according to the registration procedure specified by E.

The output z_{R} of Apply indicates whether the voter has completed registration. Every intermediate invocation returns $z_{\mathsf{R}} = \bot$, whereas the terminal invocation returns

$$\mathsf{req}_{\mathsf{R}} = \bot, \qquad z_{\mathsf{R}} = \mathsf{cred}_i.$$

Here, cred_i denotes the voter's resulting credential, which may consist of secret keys, authentication material, or other protocol-specific information required for voting.

For protocols requiring no voter-generated registration message, the algorithm Apply is not used except for receiving the credential from registrar, and just a single Register invocation may return all credential-issuance information, and the subsequent Apply invocation outputs cred_i.

Authorization and voting. To prepare one ballot, voter i runs

$$(\mathsf{req}, \omega) \leftarrow \mathsf{Request}(\mathsf{pp}, \mathsf{rst}, i, \mathsf{cred}_i, m; r_{\mathsf{Q}}).$$

The request req is delivered to the authorization authority, while the voter's private authorization state ω remains private to the voter. The authority then runs

$$(\mathsf{ans}, \mathsf{rd}', \mathsf{rst}') \leftarrow \mathsf{Authorize}(\mathsf{pp}, \mathsf{rd}, \mathsf{rst}, i, \mathsf{req}; r_{\mathsf{A}})$$

and applies the eligibility, usage, duplicate-request, and revoting policies specified by E. Finally, the voter computes

$$\beta \leftarrow \mathsf{Vote}(\mathsf{pp}, \mathsf{cred}_i, m, \omega, \mathsf{ans}; r_{\mathsf{V}}).$$

The notation r_{V} denotes the complete local tape retained for individual verification. In addition to the fresh coins used by Vote, it may contain protocol-prescribed values copied from ω. Such values remain local to the voter and are not included in the ballot unless the protocol explicitly prescribes otherwise.

For a protocol requiring no online authorization, use

$$\mathsf{Request}(\mathsf{pp}, \mathsf{rst}, i, \mathsf{cred}_i, m; r_{\mathsf{Q}}) = (\bot, \bot),$$
$$\mathsf{Authorize}(\mathsf{pp}, \mathsf{rd}, \mathsf{rst}, i, \bot; r_{\mathsf{A}}) = (\bot, \mathsf{rd}, \mathsf{rst}).$$

Validation. The validation server or the bulletin board administrator processes submitted ballots and protocol-prescribed authority events, checking their validity and updating the internal ballot box. Its input belongs to $\mathcal{X}_{\mathsf{E}} = \mathcal{A}_{\mathsf{bal}} \uplus \mathcal{T}_{\mathsf{E}}$, where $\mathcal{A}_{\mathsf{bal}}$ is the ballot space and $\mathcal{T}_{\mathsf{E}}$ contains authority-activation events, such as generating auxiliary ballots. Initially, $\mathsf{Records}(\mathsf{IBB}_0) = \epsilon$. For

$$(b_{\mathsf{val}}, \mathsf{BBd}', \mathsf{IBB}') \leftarrow \mathsf{Valid}(\mathsf{pp}, \mathsf{BBd}, \mathsf{rst}, \mathsf{IBB}, x),$$

b_{val} indicates acceptance, BBd' is the updated server state, and IBB' is the updated internal ballot box. The update appends a sequence of records $\Delta \in \mathcal{A}^*_{\mathsf{rec}}$, satisfying

$$\mathsf{Records}(\mathsf{IBB}') = \mathsf{Records}(\mathsf{IBB}) \| \Delta.$$

For a submitted ballot $x = \beta$, acceptance requires recording the ballot, whereas rejection appends nothing:

$$b_{\mathsf{val}} = 1 \implies \mathsf{rec}(\beta) \in \Delta,$$
$$b_{\mathsf{val}} = 0 \implies \Delta = \epsilon.$$

An accepted ballot may also trigger updates to the ballot-processing metadata μ_{bal}, such as its verification status or latest-ballot indicator, and the insertion of auxiliary, cover, twin, or noise records.

For an authority-activation event $x \in \mathcal{T}_{\mathsf{E}}$, we set $b_{\mathsf{val}} = 1$. Such an event performs protocol-prescribed processing without a voter-submitted ballot, possibly appending additional authorized records or leaving the record sequence unchanged.

Publication and individual verification. The algorithm

$$\mathsf{PBB} \leftarrow \mathsf{Publish}(\mathsf{pp}, \mathsf{rst}, \mathsf{IBB})$$

exposes the protocol-prescribed public view of the internal ballot box through $\mathcal{F}_{\mathsf{BB}}[\mathsf{E}]$. The algorithm may be invoked at any point in time, providing access to the same stateful bulletin board with the public information available at that prefix.

To verify one's own ballot β, voter i runs

$$b_{\mathsf{vote}} \leftarrow \mathsf{VerifyVote}(\mathsf{pp}, \mathsf{cred}_i, m, r_{\mathsf{V}}, \beta, \mathsf{PBB}).$$

Using the credential cred_i, intended choice m, retained local information r_{V}, and ballot β, the voter derives a protocol-specific verification index, such as a ballot identifier or

tracker. The algorithm may use this index to retrieve the corresponding record through the selective-read interface of PBB. If the index is invalid, no matching record exists, or the verification policy denies access, the read returns $\perp$ and $b_{\mathsf{vote}} = 0$. Otherwise, VerifyVote performs the protocol-prescribed cast-as-intended, recorded-as-cast, or inclusion checks and returns $b_{\mathsf{vote}} = 1$ exactly when they succeed.

Tallying. After voting closes, the tallier(s) run

$$(R, \pi) \leftarrow \mathsf{Tally}(\mathsf{pp}, \mathsf{td}, \mathsf{rst}, \mathsf{IBB})$$

to compute the election result R from the final ballot records, using their private state td and the public registration state rst. The algorithm applies the protocol-specific counting and revoting rules, and generates proof or audit evidence π, such as decryption or mixing proofs, to support the correctness of R.

The final public bulletin board is obtained as

$$\mathsf{PBB}^{\star} \leftarrow \mathsf{Publish}(\mathsf{pp}, \mathsf{rst}, \mathsf{IBB}).$$

The evidence π must bind R to the public and audit views exposed by $\mathsf{PBB}^{\star}$. Any party can then run

$$\mathsf{VerifyTally}(\mathsf{pp}, \mathsf{PBB}^{\star}, R, \pi)$$

to verify the announced result using only public information. It returns 1 exactly when the protocol-prescribed tally relation verifies.

Correctness. Now we define the correctness of e-voting protocols.

Definition 2 (Correctness)**.** Fix an election descriptor E and any polynomial-length, policy-admissible vote sequence

$$\mathbf{q} = ((i_1, m_1), \ldots, (i_\ell, m_\ell)), \quad i_j \in \mathcal{U},\ m_j \in \mathcal{M}.$$

Let τ be the random trace of an honest execution of $\mathbf{q}$ with any execution order permitted by E. For each ballot β_j, let $b_{\mathsf{val},j}$ and Δ_j denote its validation result and appended records. Let $\mathcal{V}(\tau)$ be the set of honest verification indices (j, k), with corresponding outputs $b_{\mathsf{vote},j,k}$. Let $\mathsf{PBB}^{\star}$ and (R, π) be the final publication and tally outputs. Let $R^{\star}_{\mathsf{E}}(\tau)$ denote the result obtained by applying the counting policy specified by E to the ballots cast in τ, counting exactly those ballots prescribed by E, independently of which ballots are actually recorded or counted by the protocol.

We say that Π_{voting} is *correct* if, for every such execution,

$$\Pr\left[\begin{array}{l} \bigwedge_{j=1}^{\ell} \left(b_{\mathsf{val},j} = 1 \wedge \mathsf{rec}(\beta_j) \in \Delta_j\right) \\ \wedge \bigwedge_{(j,k)\in\mathcal{V}(\tau)} (b_{\mathsf{vote},j,k} = 1) \\ \wedge\, \mathsf{VerifyTally}(\mathsf{pp}, \mathsf{PBB}^{\star}, R, \pi) = 1 \\ \wedge\, R = R^{\star}_{\mathsf{E}}(\tau) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda),$$

where the probability is over all protocol and bulletin-board randomness.

Correctness ensures that an honestly executed election produces the intended result. Specifically, every honestly submitted ballot is accepted and recorded, all prescribed honest individual verifications succeed, and the announced result passes public verification. Moreover, the result must agree with the voters' intended choices under the election's counting and revoting rules, even when voting sessions are interleaved or voters cast multiple ballots.

2.2. Verifiability

We adapt the weak verifiability [55] and strong verifiability [56] notions of Cortier et al., which are standard notions of verifiability of e-voting protocols, to our unified syntax.

2.2.1. Weak verifiability. Fix an election descriptor E. Let $\mathcal{R}_{\mathsf{E}}$ be its result space, and let ρ_{E} map each admissible *effective vote profile* $H \subseteq \mathcal{U} \times \mathcal{M}$ to its election result, where H contains at most one pair (i, m) for each $i \in \mathcal{U}$. The function ρ_{E} is determined by the counting policy of E. Fix also an additional trust predicate $\psi(\tau)$, specified before setup, on corruptions and channel assumptions (e.g., an honest tallying threshold). It may not depend on vote contents or the announced result.

In the following experiment, the registration and authorization authorities, ballot-validation server, and bulletin-board publication procedure are honest. The adversary $\mathcal{A}$ may corrupt tallier(s) and may adaptively corrupt voters, provided no trusted role is corrupted (including through a party holding multiple roles). Uncorrupted talliers execute their prescribed protocol roles. The challenger maintains the registration state $(\mathsf{rd}, \mathsf{rst})$, validation state BBd, internal bulletin board IBB, and an execution trace τ. The adversary accesses the following interfaces, all subject to the policies specified by E:

- $\mathcal{O}_{\mathsf{reg}}$: executes the honest Apply–Register interaction, updates $(\mathsf{rd}, \mathsf{rst})$, and delivers cred_i to the voter as prescribed by the registration policy; $\mathcal{A}$ sees the prescribed public outputs.
- $\mathcal{O}_{\mathsf{auth}}(i, \mathsf{req})$: invokes $\mathsf{Authorize}(\mathsf{pp}, \mathsf{rd}, \mathsf{rst}, i, \mathsf{req}; r_{\mathsf{A}})$, returns ans, and updates $(\mathsf{rd}, \mathsf{rst})$.
- $\mathcal{O}_{\mathsf{vote}}(i, m)$: for an uncorrupted, registered voter i and $m \in \mathcal{M}$, executes the honest Request, Authorize, and Vote algorithms and submits the resulting validation event x to Valid. Successful submissions are recorded in τ with their voter, choice, and order of occurrence.
- $\mathcal{O}_{\mathsf{cast}}(x)$: submits an arbitrary event x to the honest validation server, which runs $\mathsf{Valid}(\mathsf{pp}, \mathsf{BBd}, \mathsf{rst}, \mathsf{IBB}, x)$ and applies its returned state updates.
- $\mathcal{O}_{\mathsf{corrupt}}(i)$: adds i to the corrupted-voter set C and discloses that voter's private state, including cred_i if available.
- $\mathcal{O}_{\mathsf{corruptT}}(p)$: corrupts a party holding a tallier role, discloses its private state, and, if p is a voter, adds it to C; it rejects parties holding a role required to remain honest.

- $\mathcal{O}_{\mathsf{honestTally}}$: runs the prescribed tallying operations involving uncorrupted talliers and exposes only their protocol-prescribed outputs.

Write $\mathcal{O}$ for the collection of these seven interfaces. All registration, authorization, and validation calls share the same mutable state; $\mathcal{A}$ observes only information exposed by the public-state and bulletin-board access policies. The honest $\mathcal{O}_{\mathsf{vote}}$ procedure does not require the voter to invoke VerifyVote.

Let C_τ be the set of voters corrupted by the end of τ. For $i \notin C_\tau$, write $\mathsf{eff}_{\mathsf{E}}(\tau, i) \in \mathcal{M} \cup \{\bot\}$ for the effective choice selected from i's successfully validated honest submissions by the revoting and counting policies of E; $\bot$ means that no such choice is counted. Define

$$H_{\mathsf{E}}(\tau, C_\tau) = \big\{(i, \mathsf{eff}_{\mathsf{E}}(\tau, i)) : i \in \mathcal{U} \setminus C_\tau,\ \mathsf{eff}_{\mathsf{E}}(\tau, i) \neq \bot\big\}.$$

Let $\mathcal{B}(C)$ denote the set of vote profiles supported on subsets of C, namely

$$\mathcal{B}(C) = \big\{\{(i, m_i) : i \in S\} : S \subseteq C,\ (m_i)_{i \in S} \in \mathcal{M}^S\big\}.$$

The set of admissible election results is

$$\mathcal{G}_{\mathsf{E}}(\tau, C_\tau) = \big\{\rho_{\mathsf{E}}(H_{\mathsf{E}}(\tau, C_\tau) \cup B) : B \in \mathcal{B}(C_\tau)\big\}.$$

Thus every uncorrupted voter's effective choice is fixed, whereas at most one arbitrary admissible choice is assigned to each corrupted voter.

Definition 3 (Weak verifiability)**.** For a PPT adversary $\mathcal{A}$, define $\mathsf{Exp}^{\mathsf{wv}}_{\Pi_{\mathsf{voting}},\mathsf{E}}(\mathcal{A}; 1^\lambda)$ as follows:

$$\begin{aligned}
(\mathsf{pp}, \mathsf{td}, \mathsf{rd}, \mathsf{BBd}) &\leftarrow \mathsf{Setup}(1^\lambda, \mathsf{E});\\
(\mathsf{rst}, \mathsf{IBB}) &\leftarrow (\mathsf{rst}_0, \emptyset);\\
\sigma &\leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{pp});\\
\mathsf{PBB} &\leftarrow \mathsf{Publish}(\mathsf{pp}, \mathsf{rst}, \mathsf{IBB});\\
(R, \pi) &\leftarrow \mathcal{A}(\sigma, \mathsf{PBB});\\
b_{\mathsf{tal}} &\leftarrow \mathsf{VerifyTally}(\mathsf{pp}, \mathsf{PBB}, R, \pi);\\
b &\leftarrow [R \neq \bot \wedge b_{\mathsf{tal}} = 1];\\
&\text{return } b \wedge \psi(\tau) \wedge [R \notin \mathcal{G}_{\mathsf{E}}(\tau, C_\tau)].
\end{aligned}$$

Here rst_0 is the protocol-prescribed initial public state; τ is the complete oracle trace. The adversary obtains only corrupted talliers' private states through $\mathcal{O}_{\mathsf{corruptT}}$; with $\psi \equiv 1$, it may corrupt all talliers, as in [55]. We say that Π_{voting} satisfies *weak verifiability under* ψ if, for every admissible polynomial-size descriptor E and every PPT adversary $\mathcal{A}$,

$$\Pr\Big[\mathsf{Exp}^{\mathsf{wv}}_{\Pi_{\mathsf{voting}},\mathsf{E}}(\mathcal{A}; 1^\lambda) = 1\Big] \le \mathrm{negl}(\lambda).$$

2.2.2. Strong verifiability. Fix an election descriptor E. For a PPT adversary $\mathcal{A}$, let $\tau \leftarrow \mathsf{Run}_{\Pi_{\mathsf{voting}},\mathsf{E}}(\mathcal{A}; 1^\lambda)$ be an execution under E, with honest setup and all subsequent interactions and corruptions governed by its policies. Let C_τ be the voters corrupted during τ. We reuse ρ_{E} and $\mathcal{B}(C)$ from weak verifiability.

For each $i \notin C_\tau$, let $h_i(\tau)$ be the ordered history of its genuine voting instructions, including rule-relevant timing, and define

$$m_i(\tau) := \mathsf{Resolve}_{\mathsf{E}}(h_i(\tau)) \in \mathcal{M} \cup \{\mathsf{none}\},$$

where $\mathsf{none} \notin \mathcal{M}$ denotes no effective vote. The function $\mathsf{Resolve}_{\mathsf{E}}$ is fixed by E, independently of the actual ballot records and tally result. Let $H_{\mathsf{E}}(\tau)$ collect these histories, and let $P_{\mathsf{E}}(H')$ be the effective profile obtained by applying $\mathsf{Resolve}_{\mathsf{E}}$ to an order-preserving subhistory $H' \preceq H_{\mathsf{E}}(\tau)$.

For each $i \notin C_\tau$, let $V_i(\tau)$ be the set of its actual VerifyVote invocations, with output bits b_q. The verification policy of E fixes $\mathfrak{V}_{\mathsf{E},i}(\tau) \subseteq 2^{V_i(\tau)} \setminus \{\emptyset\}$, whose members are sets of checks constituting complete verification of $m_i(\tau)$. Membership depends on the voting history and the checks' scopes and times, not their outputs or the tally result. The family is empty if no complete verification was performed; a complete verification procedure must be available for every nonempty effective choice in an all-honest execution.

Define the successfully verified honest profile

$$Q_{\mathsf{E}}(\tau) = \{\,(i, m_i(\tau)) : i \notin C_\tau,\ m_i(\tau) \neq \mathsf{none},\ \exists S \in \mathfrak{V}_{\mathsf{E},i}(\tau),\ \forall q \in S,\ b_q = 1\,\}.$$

Let $\mathsf{RetainOK}_{\mathsf{E}}(\tau, H')$ be the predicate fixed by E before setup that permits only policy-authorized omissions without altering or fabricating instructions, and preserves successful verification unless superseded by a later genuine instruction. The admissible election results are

$$\begin{aligned}
\mathcal{G}^{\mathsf{sv}}_{\mathsf{E}}(\tau) = \big\{\, \rho_{\mathsf{E}}(P_{\mathsf{E}}(H') \cup B) :\ & H' \preceq H_{\mathsf{E}}(\tau),\\
& \mathsf{RetainOK}_{\mathsf{E}}(\tau, H') = 1,\\
& Q_{\mathsf{E}}(\tau) \subseteq P_{\mathsf{E}}(H'),\\
& B \in \mathcal{B}(C_\tau)\,\big\}.
\end{aligned}$$

Let $\mathcal{P}_{\mathsf{R}}$ and $\mathcal{P}_{\mathsf{B}}$ be the parties holding registration/authorization and validation/publication roles, respectively. Write $\mathrm{Hon}_\tau(S)$ if no party in S is corrupted during τ, and define

$$\phi_{\mathsf{sv}}(\tau) = \big(\mathrm{Hon}_\tau(\mathcal{P}_{\mathsf{R}}) \vee \mathrm{Hon}_\tau(\mathcal{P}_{\mathsf{B}})\big) \wedge \psi(\tau).$$

Let $(\mathsf{PBB}_\tau, R_\tau, \pi_\tau)$ be the final public output, and define

$$\mathsf{Acc}(\tau) = [R_\tau \in \mathcal{R}_{\mathsf{E}} \setminus \{\bot\} \wedge \mathsf{VerifyTally}(\mathsf{pp}, \mathsf{PBB}_\tau, R_\tau, \pi_\tau) = 1].$$

Definition 4 (Strong verifiability under ψ)**.** Let $\delta_{\mathsf{v}} \in [0, 1]$. The protocol Π_{voting} is *strongly verifiable under* ψ *with tolerance* δ_{v} if, for every supported polynomial-size family of election descriptors E and every PPT adversary $\mathcal{A}$, there exists a negligible function ν such that

$$\Pr\begin{bmatrix}\phi_{\mathsf{sv}}(\tau) = 1\\ \wedge\ \mathsf{Acc}(\tau) = 1\\ \wedge\ R_\tau \notin \mathcal{G}^{\mathsf{sv}}_{\mathsf{E}}(\tau)\end{bmatrix} \le \delta_{\mathsf{v}} + \nu(\lambda),$$

where $\tau \leftarrow \mathsf{Run}_{\Pi_{\mathsf{voting}},\mathsf{E}}(\mathcal{A}; 1^\lambda)$. We write simply *strong verifiability under* ψ when $\delta_{\mathsf{v}} = 0$.

Intuitively, strong verifiability ensures that an accepted tally preserves all successfully verified honest votes, allows only policy-permitted omissions of other honest votes, and counts at most one effective choice per corrupted voter.

This guarantee holds whenever either the registration/authorization parties or the validation/publication parties remain honest, subject to ψ.

2.3. Ballot privacy

We adapt BPRIV [57] to our interactive syntax. Fix an election descriptor E. All authorities are honest; before setup, the adversary chooses $C \subseteq \mathcal{U}$ of corrupted voters, excluding parties holding trusted authority roles. It obtains their private states and messages, together with all information publicly accessible under E.

For $b \in \{0,1\}$, the challenger runs Setup and maintains two copies of all mutable authority, board, and honest-voter states, with common initial election keys and the prescribed initialization. Only copy b is observable. Corresponding calls use fresh random tapes, shared when their inputs and accessed local states coincide, and independent otherwise. Calls share persistent state within each copy and follow the interaction policies of E.

For $s_j = (\mathsf{BBd}^{(j)}, \mathsf{IBB}^{(j)})$, let $\mathsf{Check}_{\mathsf{E}}$ and $\mathsf{Update}_{\mathsf{E}}$ denote the acceptance test and state-update parts of Valid. The update for either decision is fixed by the protocol before the game. For paired inputs (x_0, x_1) and validation tapes (r_0, r_1), execute

$$a \leftarrow \mathsf{Check}_{\mathsf{E}}(\mathsf{pp}, \mathsf{rst}^{(b)}, s_b, x_b; r_b),$$

$$s_j \leftarrow \mathsf{Update}_{\mathsf{E}}(\mathsf{pp}, \mathsf{rst}^{(j)}, s_j, x_j, a; r_j), \quad j \in \{0,1\}.$$

Both maps use only their displayed inputs; the hidden copy's acceptance test is not rerun.

The adversary accesses the following interfaces, polynomially many times in any order permitted by E:

- $\mathcal{O}_{\mathsf{voteLR}}(i, m_0, m_1)$, for $i \in \mathcal{U} \setminus C$ and $m_0, m_1 \in \mathcal{M}$, starts paired voting sessions with identical voter identity, session metadata, and public control inputs. Copy j uses m_j in every choice-dependent call. Only a session handle is returned; credentials, random tapes, and honest session states remain private.
- $\mathcal{O}_{\mathsf{step}}(q)$ executes the operation specified by q, including registration, authorization, voting, or individual verification, in both copies. Session handles identify corresponding local sessions. Honest messages are computed separately; adversarial messages are supplied unchanged to both copies. Validation follows the paired rule, and only the prescribed outputs of copy b are exposed.
- $\mathcal{O}_{\mathsf{cast}}(x)$ submits an arbitrary validation event using $x_0 = x_1 = x$. Copied events are permitted; authentication and replay checks are performed by Valid.
- $\mathcal{O}_{\mathsf{read}}$ returns the accessible public state and board view of copy b, including $\mathsf{Publish}(\mathsf{pp}, \mathsf{rst}^{(b)}, \mathsf{IBB}^{(b)})$.
- $\mathcal{O}_{\mathsf{tally}}$ closes the election and returns the tally output defined below. It may be called only once; afterwards, the adversary outputs its bit using only local computation.

Repeated voting sessions are permitted according to E. Failures, rejections, and their state updates follow the actual session code and are retained in the experiment.

At tallying, the challenger freezes both copies and computes

$$(R, \pi_0) \leftarrow \mathsf{Tally}(\mathsf{pp}, \mathsf{td}, \mathsf{rst}^{(0)}, \mathsf{IBB}^{(0)}).$$

It returns (R, π_0) if $b = 0$, or (R, π_1) if $b = 1$, where

$$\pi_1 \leftarrow \mathsf{SimTally}(\mathsf{pp}, \mathsf{rst}^{(1)}, \mathsf{IBB}^{(1)}, V_1, R).$$

Here V_1 is copy 1's observable pre-tally transcript, excluding challenge choices and honest-voter private states; π includes all public tally auxiliary data. The simulator receives only the displayed inputs and the simulation access permitted by the fixed setup model. When required, the reference computation uses the real global setup and the alternative displayed computation uses its simulated counterpart, with only the prescribed trapdoor or oracle-programming access [57, Section IV-A]. The adversary may run $\mathsf{VerifyTally}$; invalid results and failed checks do not exclude a run.

Definition 5 (Ballot privacy)**.** Let $\mathsf{Exp}^{\mathsf{bpriv},b}_{\Pi_{\mathsf{voting}},\mathsf{E},C}(\mathcal{A}, \mathsf{SimTally}; 1^\lambda)$ be the adversary's output bit in the experiment above. The protocol Π_{voting} has *ballot privacy* if, for every supported polynomial-size descriptor family E, there exists a PPT simulator $\mathsf{SimTally}$ such that, for every permitted C and PPT adversary $\mathcal{A}$,

$$\left| \begin{array}{l} \Pr[\mathsf{Exp}^{\mathsf{bpriv},0}_{\Pi_{\mathsf{voting}},\mathsf{E},C}(\mathcal{A}, \mathsf{SimTally}; 1^\lambda) = 1] \\ -\Pr[\mathsf{Exp}^{\mathsf{bpriv},1}_{\Pi_{\mathsf{voting}},\mathsf{E},C}(\mathcal{A}, \mathsf{SimTally}; 1^\lambda) = 1] \end{array} \right| \leq \mathrm{negl}(\lambda).$$

The probability includes setup, protocol, adversarial, and simulation randomness.

The adversary must not distinguish the two visible executions when both experiments tally copy 0 and the alternative tally evidence is simulated. No equality between the two copies' tally results is required.

Election-rule consistency. Separately, we adapt the strong-consistency requirement of [57, Section IV-B] to voting-action records. Let $\mathcal{D}_{\mathsf{E}}$ be the space of plaintext action records, containing the identifiers, choices, authorization information, and operation labels required by E, including a designated invalid-record symbol. The fixed, efficiently computable function

$$F_{\mathsf{E}} : \mathcal{D}_{\mathsf{E}}^* \longrightarrow \mathcal{R}_{\mathsf{E}} \cup \{\perp\}$$

applies E's election policies to an ordered history, independently of cryptographic encodings and Tally.

In an ordinary execution with honest authorities, let L be the canonical log of accepted voting-action records, retaining entries despite later board updates. Each record contains only that action's messages and fixed authentication and public-control context. Let κ contain the prescribed authority decoding keys, excluding vote-dependent history. The protocol satisfies *election-rule consistency* if there exists a deterministic PPT extractor $\mathsf{Extract}(\mathsf{pp}, \kappa, a) \in \mathcal{D}_{\mathsf{E}}$ using only these inputs, such that extraction recovers the prescribed plaintext of every honestly generated action record except with negligible probability, and every PPT adversary controlling voters satisfies

$$\Pr\left[\begin{array}{c} \text{a tally is released} \wedge \\ R \neq F_{\mathsf{E}}\big(\mathsf{Extract}^*(L)\big) \end{array} \right] \leq \mathrm{negl}(\lambda).$$

Here $(R, \pi) \leftarrow \mathsf{Tally}(\mathsf{pp}, \mathsf{td}, \mathsf{rst}, \mathsf{IBB})$, and $\mathsf{Extract}^*$ applies extraction entrywise, preserving order and multiplicity.

This separate condition requires the released result to follow E on the decoded action history; it is not assumed by the ballot-privacy experiment.

3. Coercion Resistance

We retain the KTV's definition of coercion resistance [3]: a counterstrategy achieving the voter's goal and is difficulty to distinguish from obedience. To compare protocols under the same assumptions, we use the common syntax of Section 2.1 and explicitly specify which principals the coercer may control and when it may contact the voter.

3.1. Definition

Common execution. Fix an election descriptor E, a single target voter $i^* \in \mathcal{U}$, and its vote choice $m^* \in \mathcal{M}$. Let $\mathcal{P}$ be the set of parties participating in the election. We assume that if a coercer corrupts a party, then its every role in the election is corrupted. Furthermore, we fix, for all comparisons, an efficiently samplable distribution of the other voters' voting histories and the adversary's auxiliary input, and the number of executions of algorithms in a security game.

The game executes the existing algorithms and delivers their messages using persistent local states, applying the same ownership and corruption rules from setup onwards. It permits repeated and interleaved sessions according to their actual state transitions. Setup, admission, publication, tallying, and all outputs produced during the fixed polynomially bounded number of algorithm executions remain part of the execution. Responses, rejections, and aborts are retained.

Attack schedule. A coercer capability Ω, determined by public information, specifies a schedule and a global number of coercions k_Ω:

$$\Omega(t) = (\mathcal{C}_t, w_t), \qquad \mathcal{C}_t \subseteq \mathcal{P} \setminus \{i^*\}, \quad w_t \in \{0, 1\}.$$

Here t counts completed algorithm calls, starting at 0 before setup; $\Omega(t)$ may depend only on t and previously released public information, not on private states. We identify i^* with the target voter's party in $\mathcal{P}$. For each t, $\mathcal{C}_t$ is the set of parties eligible for new corruption at time t, and these sets need not be nondecreasing: at time t, the adversary may corrupt any previously uncorrupted party in $\mathcal{C}_t$, based on its accumulated information, obtaining its current local state and controlling its subsequent computations and interfaces. This includes a coercer acting as a participating party itself.

Interactive communications between the coercer and a fixed coerced party i^* may be initiated only if $w_t = 1$, with at most one direct coercion exchange per time and at most k_Ω exchanges throughout the execution. At such a contact, the adversary may request every information the voter has, and supply an instruction, which is an arbitrary PPT algorithm for voter to do, allowing to preserve the voter's internal state. Execution of an instruction may continue locally after contact ends.

Information and voter behavior. The common syntax fixes which values are public and which are held privately by their respective parties for all protocols. The adversary obtains every released public value, including $\mathsf{pp}, \mathsf{rst}, \mathsf{PBB}, R, \pi$, and may query every public interface. It also obtains the actual states and messages sent or received by corrupted parties, and the reports returned by the target. Messages are public unless the common syntax specifies private delivery. Private values remain with their owners unless exposed by these rules. A protocol must follow these rules; it cannot hide a public message by replacing it with $\perp$. Unused fields are also denoted by $\perp$. All information already obtained persists.

Before the first coercion, the target voter follows its honest program with input m^* in both worlds. After the start of the first coercion, the target continues from its retained state using either an instruction Obey contained in the coercion or a PPT counterstrategy Res. A counterstrategy Res may deviate and report fake information, but this counterstrategy can be made only by the target's actual local state and interfaces. Thus a demand for a state report is not a direct read of the target's genuine memory. Neither strategy receives other party's private state from the experiment. This convention compares continuations under the same honest behavior before the first contact.

Experiment and goal. For $b \in \{0, 1\}$, the experiment $\mathsf{Exp}^b_{\Pi_{\mathsf{voting}},\Omega}(\mathcal{A}, \mathsf{Res}; 1^\lambda)$ initializes the election and executes its algorithm calls and message deliveries as above, enforcing Ω throughout. At the first coercion it uses Obey if $b = 0$, or Res if $b = 1$. The adversary adaptively chooses its actions from its accumulated information and, after the execution, outputs a bit d_b. Let τ_b be the record of every execution in world b during the security experiment. The experiment outputs d_b. Both worlds use the same input and initialization distributions.

For the vote goal, define γ_{i^*,m^*} as a binary function on complete execution records $\tau = (\sigma_0, e_1, \sigma_1, \ldots, e_L, \sigma_L)$, where σ_t records the states of all parties and the bulletin board after t algorithm calls, and e_t records the t-th call together with the intervening messages and adversarial actions:

$$\gamma_{i^*,m^*}(\tau) = 1 \iff \text{a ballot cast by } i^* \text{ for } m^* \text{ is selected and counted in the announced tally } R.$$

γ_{i^*,m^*} is evaluated on the full record using the protocol's ballot-selection and counting rules. If the ballot is not counted or no tally is announced, then $\gamma_{i^*,m^*}(\tau) = 0$. Probabilities include input sampling and all execution randomness; no run is discarded or conditioned on successful completion or matching tallies.

Definition 6 (Schedule-dependent coercion resistance)**.** Fix Ω and a bound $\delta \in [0, 1]$.

Fix a target voter $i^* \in \mathcal{U}$ and a voter's choice $m^* \in \mathcal{M}$. Let $L = L(\lambda)$ be the fixed, polynomially bounded number of algorithm executions, and let $t \in \{0, \ldots, L-1\}$ denote the time before the $(t+1)$-st execution.

Experiment $\mathsf{Exp}^{\mathsf{CR}}_{\Pi_{\mathsf{voting}},\Omega}(\mathcal{A}, \mathsf{Res}; 1^\lambda, i^*, m^*)$

(1) $b \xleftarrow{\$} \{0,1\}$; sample the other voters' voting histories and the adversary's auxiliary input from the fixed distribution.
(2) Set $\mathsf{mode} \leftarrow \mathsf{Honest}$ and $c \leftarrow 0$.
(3) **for** $t = 0, \ldots, L-1$ **do**
(4) Allow $\mathcal{A}$ to corrupt parties in $\mathcal{C}_t$ and control corrupted parties.
(5) **if** $w_t = 1$, $c < k_\Omega$, and $\mathcal{A}$ requests direct contact **then**
(6) **on first contact:** $\mathsf{mode} \leftarrow \begin{cases} \mathsf{Obey}, & b = 0, \\ \mathsf{Res}, & b = 1. \end{cases}$
(7) $\mathcal{A}$ coerces i^* according to mode; set $c \leftarrow c + 1$.
(8) Execute the next algorithm call of Π_{voting}, starting with $\mathsf{Setup}(1^\lambda, \mathsf{E})$, and process messages according to the common syntax and E.
(9) Give $\mathcal{A}$ all information accessible under the common syntax and Ω.
(10) **end for**
(11) Let τ_b be the complete execution record; obtain $b' \in \{0,1\}$ from $\mathcal{A}$.
(12) **return** 1 if $b' = b$, and 0 otherwise.

Figure 2. Schedule-dependent coercion-resistance experiment.

A coercer capability is $\Omega = ((\mathcal{C}_t, w_t)_{t=0}^{L-1}, k_\Omega)$, where $\mathcal{C}_t \subseteq \mathcal{P} \setminus \{i^*\}$ is the set of parties that may be newly corrupted at time t, $w_t \in \{0,1\}$ indicates whether direct contact with i^* is permitted at time t, and $k_\Omega \geq 0$ is a polynomially bounded integer. We write $\Omega(t) = (\mathcal{C}_t, w_t)$. Corruption is permanent and grants control over all roles and retained states of the corrupted party. An adversary respects Ω if it corrupts parties only when permitted and initiates at most one direct coercion exchange at each time t with $w_t = 1$, and at most k_Ω exchanges throughout the execution. Let Obey be the obedient voter strategy that follows the adversary's instructions and truthfully reports its retained information, continuing its current program when no new instruction is given.

For PPT algorithms $\mathcal{A}$ and Res, let $\mathsf{Exp}^{\mathsf{CR}}_{\Pi_{\mathsf{voting}},\Omega}(\mathcal{A}, \mathsf{Res}; 1^\lambda, i^*, m^*)$ be the experiment in Figure 2. The probabilities below are taken over this experiment. Here $b \in \{0,1\}$ is the uniformly sampled experiment bit, b' is the adversary's output, and τ_b is the complete execution record in world b, where $b = 0$ uses Obey and $b = 1$ uses Res.

We say that an e-voting protocol Π_{voting} satisfies $\mathsf{CR}^\Omega(\gamma_{i^*,m^*}, \delta)$-coercion resistance if there exists a PPT counterstrategy Res such that, for every PPT adversary $\mathcal{A}$ respecting Ω, there is a negligible function η for which

$$\Pr[\gamma_{i^*,m^*}(\tau_b) = 1 \mid b = 1] \geq 1 - \eta(\lambda), \tag{1}$$

$$\left| \Pr[b' = b] - \frac{1}{2} \right| \leq \frac{\delta}{2} + \eta(\lambda). \tag{2}$$

The same Res must satisfy both conditions. It may use public information and its own local state, but not the adversary's undisclosed state. For a claim covering all voters and choices, Res takes (i^*, m^*) as input and the conditions hold for every such pair.

Condition (1) requires effective resistance, whereas (2) hides that resistance from the coercer. The bound δ is fixed before analysis and allows for information in the public election result, as in KTV [3]. Our refinement makes role control and contact timing explicit while preserving one continuous execution and its entire observable history.

3.2. Classification of coercion settings

Now we classify how Ω in Definition 6 can be decomposed to settings for the experiment.

Fix a target voter $i^* \in \mathcal{U}$ and a choice $m^* \in \mathcal{M}$. A corruption bound is a set $C \subseteq \mathcal{P} \setminus \{i^*\}$ of parties that the adversary may corrupt. Let $\rho \in \{\mathsf{stat}, \mathsf{adapt}\}$ be the corruption mode. If $\rho = \mathsf{stat}$, the adversary may choose any subset of C before setup, but cannot corrupt additional parties thereafter; if $\rho = \mathsf{adapt}$, it may corrupt previously uncorrupted parties in C at any time, based on its current view. Corruption is permanent. Let $k \geq 0$ be an integer bounded by a polynomial in λ.

Let R, V, and I denote registration ($\mathsf{Apply}, \mathsf{Register}$), voting ($\mathsf{Request}, \mathsf{Authorize}, \mathsf{Vote}$ and ballot submission), and individual verification ($\mathsf{VerifyVote}$) phases, respectively. For $X \in \{\mathsf{R}, \mathsf{V}, \mathsf{I}\}$, let $J_X \subseteq \{0, \ldots, L-1\}$ be the publicly specified coercion times associated with phase X. For nonempty $B \subseteq \{\mathsf{R}, \mathsf{V}, \mathsf{I}\}$, put $J_B = \bigcup_{X \in B} J_X$ and write $J_B = \{t_1 < \cdots < t_s\}$. Coercion permission is determined from public timing and phase information, without access to the hidden experiment bit or private states. The phases need not occur in disjoint or consecutive intervals.

Let $\tau \in \{0, \ldots, L-1\}$ be a publicly specified cutoff time. For $J_B = \{t_1 < \cdots < t_s\}$, define three time models $W \in \{\mathsf{init}, \mathsf{pre}(\tau), \mathsf{full}\}$ with corresponding contact windows:

$$J_{B,\mathsf{init}} := \{t_j : 1 \leq j \leq \min(k, s)\},$$
$$J_{B,\mathsf{pre}(\tau)} := \{t \in J_B : t \leq \tau\},$$
$$J_{B,\mathsf{full}} := J_B.$$

In each model, the adversary may adaptively initiate at most k direct coercion exchanges, at most one at each eligible time. The initial model init restricts coercion to the first k eligible times, the prefix model $\mathsf{pre}(\tau)$ permits any k contacts before the cutoff τ, and the full model permits any k contacts throughout the selected phases.

Definition 7 (Classified coercer capabilities)**.** For a corruption bound C, a corruption mode $\rho \in \{\mathsf{stat}, \mathsf{adapt}\}$, a nonempty coercion-phase set B, a time model W, and the number of coercions k, define the coercer capability

$$\Omega_{C,\rho,B,W,k} := \big((\mathcal{C}_t^{C,\rho}, \mathbf{1}_{J_{B,W}}(t))_{t=0}^{L-1}, k\big),$$

$$\text{where } \mathcal{C}_t^{C,\rho} := \begin{cases} C, & t = 0 \text{ or } \rho = \mathsf{adapt}, \\ \varnothing, & 1 \leq t < L \text{ and } \rho = \mathsf{stat}. \end{cases}$$

Here $J_{B,W}$ is the coercion window determined by the selected phases B and time model $W \in \{\mathsf{init}, \mathsf{pre}(\tau), \mathsf{full}\}$. Thus, the adversary may corrupt any subset of C before setup, retaining control thereafter, and may initiate at most k direct coercion exchanges at eligible times in $J_{B,W}$. If

$\rho = \mathsf{adapt}$, the adversary may additionally corrupt previously uncorrupted parties in C at later times, choosing those parties based on its current view.

For brevity, $\mathsf{CR}^{C,\rho,B,W,\le k}_{\Pi_{\mathsf{voting}}}(\gamma_{i^*,m^*},\delta)$ denotes the statement that Π_{voting} satisfies $\mathsf{CR}^{\Omega_{C,\rho,B,W,k}}(\gamma_{i^*,m^*},\delta)$ as defined in Definition 6.

Representative corruption bounds are $\varnothing$ (outsider), $\{p_{\mathsf{R}}\}$ (registrar), $\{p_{\mathsf{B}}\}$ (board/validation), and C_{T} (talliers), where $p_{\mathsf{R}}, p_{\mathsf{B}} \in \mathcal{P} \setminus \{i^*\}$ are the parties performing the indicated roles and $C_{\mathsf{T}} \subseteq \mathcal{P} \setminus \{i^*\}$ is a set of tallier parties; a general bound C may also contain other voters. If one party performs multiple roles, corrupting it compromises all those roles. Taking $B = \{\mathsf{V}\}$ gives voting-only settings, while other nonempty choices of B cover registration, individual verification, and their combinations.

TABLE 2. THE FIVE PARAMETERS SPECIFYING A COERCION SETTING. EACH ROW DESCRIBES ONE CLASSIFICATION AXIS.

Axis	Parameter	Allowed values
Corruption bound	C	$C \subseteq \mathcal{P} \setminus \{i^*\}$
Corruption mode	ρ	stat, adapt
Coercion phase	B	$\varnothing \neq B \subseteq \{\mathsf{R}, \mathsf{V}, \mathsf{I}\}$
Time model	W	init, $\mathsf{pre}(\tau)$, full
number of coercions	k	$k \ge 0$, polynomially bounded in λ

Remark 8. Appendix B extends the corruption-bound axis to threshold tallier corruption, allowing the adversary to choose a bounded number of talliers to corrupt, either statically or adaptively.

Proposition 9 (Monotonicity)**.** *For fixed $\Pi_{\mathsf{voting}}, i^*, m^*, \delta$ and common input and execution rules, let $(C_j, \rho_j, B_j, W_j, k_j)$ be two coercion settings, $j \in \{1,2\}$. If $C_1 \subseteq C_2$, $\rho_1 = \rho_2$ or $(\rho_1,\rho_2) = (\mathsf{stat},\mathsf{adapt})$, $J_{B_1,W_1} \subseteq J_{B_2,W_2}$, and $k_1 \le k_2$, then*

$$\mathsf{CR}^{C_2,\rho_2,B_2,W_2,\le k_2}_{\Pi_{\mathsf{voting}}}(\gamma_{i^*,m^*},\delta) \implies \mathsf{CR}^{C_1,\rho_1,B_1,W_1,\le k_1}_{\Pi_{\mathsf{voting}}}(\gamma_{i^*,m^*},\delta).$$

In particular, for fixed C,ρ,B,k, if $s,k \ge 1$ and $\tau \ge t_{\min(k,s)}$, then

$$\mathsf{CR}^{C,\rho,B,\mathsf{full},\le k}_{\Pi_{\mathsf{voting}}}(\gamma_{i^*,m^*},\delta) \implies \mathsf{CR}^{C,\rho,B,\mathsf{pre}(\tau),\le k}_{\Pi_{\mathsf{voting}}}(\gamma_{i^*,m^*},\delta) \implies \mathsf{CR}^{C,\rho,B,\mathsf{init},\le k}_{\Pi_{\mathsf{voting}}}(\gamma_{i^*,m^*},\delta).$$

The same implications hold for $s = 0$ or $k = 0$. For $B_1 \subseteq B_2$, the required window inclusion holds with the same full or prefix time model, but not necessarily with the same initial-k time model.

Proof. Every attack allowed by the weaker setting is also allowed by the stronger one: the adversary may select the same corruption choices at the same times and the same adaptive contact times. Thus the same counterstrategy satisfies both security conditions, with unchanged bounds. □

3.2.1. Impossible coercion setting. Let $\mathcal{P}_R \subseteq \mathcal{P} \setminus \{i^*\}$ be the nonempty set of all parties authorized to announce a tally R. Assume that an announced tally can arise only from an action of a party in $\mathcal{P}_R$; in particular, no independent functionality announces R without such an action.

Proposition 10 (Complete corruption of tally announcers)**.** *For every e-voting protocol satisfying the above announcement rule, if $\mathcal{P}_R \subseteq C$, then, for every $\rho \in \{\mathsf{stat},\mathsf{adapt}\}$, B, W, k, and $\delta \in [0,1]$, then the protocol does not satisfy $\mathsf{CR}^{\Omega_{C,\rho,B,W,k}}(\gamma_{i^*,m^*},\delta)$.*

Proof. Consider the adversary that corrupts every party in $\mathcal{P}_R$ before setup, suppresses all their tally announcements, and never coerces i^*. This is permitted under either corruption mode and any coercion restriction. Since no other party or functionality can announce R, no tally is announced. Thus $\Pr[\gamma_{i^*,m^*}(\tau_1) = 1] = 0$ for every counterstrategy, contradicting (1). □

The result is independent of the selected coercion phases, time model, and contact budget. It does not apply if an uncorrupted party or independent functionality can announce the tally. In particular, the classification parameters alone do not establish any further protocol-independent impossibility.

References

[1] A. Juels, D. Catalano, and M. Jakobsson, "Coercion-resistant electronic elections," in *Proceedings of the 2005 ACM Workshop on Privacy in the Electronic Society*, 2005, pp. 61–70.

[2] R. Kusters and T. Truderung, "An epistemic approach to coercion-resistance for electronic voting protocols," in *2009 30th IEEE Symposium on Security and Privacy*. IEEE, 2009, pp. 251–266.

[3] R. Küsters, T. Truderung, and A. Vogt, "A game-based definition of coercion resistance and its applications," *Journal of Computer Security*, vol. 20, no. 6, pp. 709–764, 2012.

[4] R. Giustolisi, M. S. Garjan, and C. Schuermann, "Thwarting last-minute voter coercion," in *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2024, pp. 3423–3439.

[5] J. Heather and S. Schneider, "A formal framework for modelling coercion resistance and receipt freeness," in *International Symposium on Formal Methods*. Springer, 2012, pp. 217–231.

[6] R. Longo, M. Mollaeefar, U. Morelli, C. Spadafora, A. Tomasi, and S. Ranise, "Modeling and assessing coercion threats in electronic voting," in *International Conference on Risks and Security of Internet and Systems*. Springer, 2024, pp. 472–488.

[7] M. R. Clarkson, S. Chong, and A. C. Myers, "Civitas: Toward a secure voting system," in *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 2008, pp. 354–368.

[8] R. Araújo, S. Foulle, and J. Traoré, "A practical and secure coercion-resistant scheme for internet voting," in *Towards Trustworthy Elections*, ser. Lecture Notes in Computer Science, vol. 6000. Springer, 2010, pp. 330–342.

[9] R. Araújo, N. Ben Rajeb, R. Robbana, J. Traoré, and S. Youssfi, "Towards practical and secure coercion-resistant electronic elections," in *Cryptology and Network Security*, ser. Lecture Notes in Computer Science, vol. 6467. Springer, 2010, pp. 278–297.

[10] M. Schläpfer, R. Haenni, R. Koenig, and O. Spycher, "Efficient vote authorization in coercion-resistant internet voting," in *E-Voting and Identity*, ser. Lecture Notes in Computer Science, vol. 7187. Springer, 2012, pp. 71–88, proceedings of Vote-ID 2011.

[11] A. Essex, J. Clark, and U. Hengartner, "Cobra: Toward concurrent ballot authorization for internet voting," in *2012 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/-WOTE 12)*. Bellevue, WA: USENIX Association, Aug. 2012.

[12] V. Cortier, P. Gaudry, and Q. Yang, "Is the jcj voting system really coercion-resistant?" in *2024 IEEE 37th Computer Security Foundations Symposium (CSF)*. IEEE, 2024, pp. 186–200.

[13] D. F. Aranha, M. Battagliola, and L. D. Roy, "Faster coercion-resistant e-voting by encrypted sorting," in *E-Vote-ID 2023*, ser. Lecture Notes in Informatics, vol. P347. Bonn: Gesellschaft für Informatik, 2024, pp. 53–70, presented at E-Vote-ID 2023.

[14] R. Giustolisi and M. S. Garjan, "Efficient cleansing in coercion-resistant voting," in *International Joint Conference on Electronic Voting*. Springer, 2024, pp. 72–88.

[15] P. Grontas, A. Pagourtzis, A. Zacharakis, and B. Zhang, "Towards everlasting privacy and efficient coercion resistance in remote electronic voting," in *International Conference on Financial Cryptography and Data Security*. Springer, 2018, pp. 210–231.

[16] S. Park, J. Choi, J. Kim, and H. Oh, "zkvoting: Zero-knowledge proof based coercion-resistant and e2e verifiable e-voting system," *Cryptology ePrint Archive*, 2024.

[17] P. Grontas, A. Pagourtzis, and M. Spyrakou, "Voting with coercion resistance and everlasting privacy using linkable ring signatures," *Cryptology ePrint Archive*, 2025.

[18] L.-H. M. J. Merino, "Votegral: Towards usable, end-to-end verifiable, coercion-resistant online voting," Ph.D. dissertation, EPFL, 2025.

[19] L.-H. Merino, S. Colombo, R. Reyes, A. Azhir, S. Mishra, P. Tennage, M. A. Raeisi, H. Zhang, J. R. Allen, B. Tellenbach *et al.*, "Trip: Coercion-resistant registration for e-voting with verifiability and usability in votegral," in *Proceedings of the ACM SIGOPS 31st Symposium on Operating Systems Principles*, 2025, pp. 834–874.

[20] O. Spycher, R. Koenig, R. Haenni, and M. Schläpfer, "A new approach towards coercion-resistant remote e-voting in linear time," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, vol. 7035. Springer, 2012, pp. 182–189, proceedings of FC 2011.

[21] R. Araújo and J. Traoré, "A practical coercion resistant voting scheme revisited," in *E-Voting and Identity*, ser. Lecture Notes in Computer Science, vol. 7985. Springer, 2013, pp. 193–209.

[22] R. Araújo, A. Barki, S. Brunet, and J. Traoré, "Remote electronic voting can be efficient, verifiable and coercion-resistant," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, vol. 9604. Springer, 2016, pp. 224–232.

[23] S. Patachi and C. Schürmann, "Eos a universal verifiable and coercion resistant voting protocol," in *Electronic Voting*, ser. Lecture Notes in Computer Science, vol. 10615. Springer, 2017, pp. 210–227.

[24] O. Kulyk, S. Neumann, K. Marky, J. Budurushi, and M. Volkamer, "Coercion-resistant proxy voting," *Computers & Security*, vol. 71, pp. 88–99, 2017.

[25] P. B. Rønne, A. Atashpendar, K. Gjøsteen, and P. Y. A. Ryan, "Coercion-resistant voting in linear time via fully homomorphic encryption: Towards a quantum-safe scheme," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, vol. 11599. Springer, 2020, pp. 289–298, proceedings of the FC 2019 Workshops.

[26] V. Iovino, A. Rial, P. B. Rønne, and P. Y. A. Ryan, "Using selene to verify your vote in jcj," in *Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, vol. 10323. Springer, 2017, pp. 385–403.

[27] D. Achenbach, C. Kempka, B. Löwe, and J. Müller-Quade, "Improved coercion-resistant electronic elections through deniable re-voting," *USENIX Journal of Election Technology and Systems (JETS)*, vol. 3, pp. 26–45, 2015.

[28] P. Locher, R. Haenni, and R. E. Koenig, "Coercion-resistant internet voting with everlasting privacy," in *International Conference on Financial Cryptography and Data Security*. Springer, 2016, pp. 161–175.

[29] State Electoral Office of Estonia, "IVXV protocols: Specification," State Electoral Office of Estonia, Technical Specification IVXV-PR-EN-1.8.0, Dec. 2022, version 1.8.0. [Online]. Available: https://www.valimised.ee/sites/default/files/2023-02/IVXV-protocols.pdf

[30] ——, "IVXV architecture: Specification," State Electoral Office of Estonia, Technical Specification IVXV-AR-EN-1.8.0, Dec. 2022, version 1.8.0. [Online]. Available: https://www.valimised.ee/sites/default/files/2023-02/IVXV-architecture.pdf

[31] W. Lueks, I. Querejeta-Azurmendi, and C. Troncoso, "Voteagain: A scalable coercion-resistant voting system," in *29th USENIX security symposium (USENIX Security 20)*, 2020, pp. 1553–1570.

[32] J. Qiao, M. Arapinis, and T. Zacharias, "On the necessity of pre-agreed secrets for thwarting last-minute coercion: Vulnerabilities and lessons from the loki e-voting protocol," *arXiv preprint arXiv:2604.00188*, 2026.

[33] J. Müller, B. Pejó, and I. Pryvalov, "Devos: deniable yet verifiable vote updating," *Proceedings on Privacy Enhancing Technologies*, 2024.

[34] L. Kimura, D. Leskow, D. F. Aranha, R. Araújo, and M. Simplicio, "Filtrumvote: Scalable, verifiable, and coercion-resistant internet voting," in *International Joint Conference on Electronic Voting*. Springer, 2026, pp. 108–124.

[35] J. Clark and U. Hengartner, "Selections: Internet voting with over-the-shoulder coercion-resistance," in *International Conference on Financial Cryptography and Data Security*. Springer, 2011, pp. 47–61.

[36] T. Haines, J. Müller, and I. Querejeta-Azurmendi, "Scalable coercion-resistant e-voting under weaker trust assumptions," in *Proceedings of the 38th ACM/SIGAPP symposium on applied computing*, 2023, pp. 1576–1584.

[37] R. Longo and C. Spadafora, "Amun: Securing e-voting against over-the-shoulder coercion," in *Proceedings of the 21st International Conference on Security and Cryptography (SECRYPT 2024)*. SciTePress, 2024, pp. 508–517.

[38] V. Farzaliyev and J. Willemson, "End-to-end verifiable internet voting with partially private bulletin boards," in *International Joint Conference on Electronic Voting*. Springer, 2025, pp. 73–89.

[39] T. Dimitriou, "Efficient, coercion-free and universally verifiable blockchain-based voting," *Computer Networks*, vol. 174, p. 107234, 2020.

[40] C. Spadafora, R. Longo, and M. Sala, "A coercion-resistant blockchain-based e-voting protocol with receipts," vol. 17, no. 2, pp. 500–521, 2023.

[41] N. D. Sarier, "Efficient, usable and coercion-resistant blockchain-based e-voting," *Journal of Information Security and Applications*, vol. 92, p. 104074, 2025.

[42] Z. Yin, B. Zhang, A. Nastenko, R. Oliynykov, and K. Ren, "A scalable coercion-resistant voting scheme for blockchain decision-making," *Cryptology ePrint Archive*, 2023.

[43] S. S. Al Qasimi, Q. Nasir, and M. A. Talib, "Zkp-e-vote: A scalable, receipt-free and coercion-resistant blockchain framework for national-level electronic elections," in *2025 International Conference on Intelligent Systems: Theories and Applications (SITA)*. IEEE, 2025, pp. 1–8.

[44] T. Finogina and J. Herranz, "On remote electronic voting with both coercion resistance and cast-as-intended verifiability," *Journal of Information Security and Applications*, vol. 76, p. 103554, 2023.

[45] ——, "Coercion-resistant cast-as-intended verifiability for computationally limited voters," in *International Conference on Financial Cryptography and Data Security*. Springer, 2023, pp. 34–46.

[46] Y.-X. Kho, S.-H. Heng, S.-Y. Tan, and J.-J. Chin, "A provably secure coercion-resistant e-voting scheme with confidentiality, anonymity, unforgeability, and cai verifiability," *Plos one*, vol. 20, no. 6, p. e0324182, 2025.

[47] A. Michalas, "Coercion-resistant voting via anamorphic encryption," in *Proceedings of the 31st ACM Symposium on Access Control Models and Technologies*. ACM, 2026, pp. 113–124.

[48] M. Backes, M. Gagné, and M. Skoruppa, "Using mobile device communication to strengthen e-voting protocols," in *Proceedings of the 12th ACM Workshop on Privacy in the Electronic Society*. ACM, 2013, pp. 237–242.

[49] S. Delaune, S. Kremer, and M. Ryan, "Verifying privacy-type properties of electronic voting protocols," *Journal of Computer Security*, vol. 17, no. 4, pp. 435–487, 2009.

[50] J. Dreier, P. Lafourcade, and Y. Lakhnech, "A formal taxonomy of privacy in voting protocols," in *2012 IEEE International Conference on Communications (ICC)*. IEEE, 2012, pp. 6710–6715.

[51] D. Unruh and J. Müller-Quade, "Universally composable incoercibility," in *Advances in Cryptology – CRYPTO 2010*, ser. Lecture Notes in Computer Science, vol. 6223. Springer, 2010, pp. 411–428.

[52] J. Alwen, R. Ostrovsky, H.-S. Zhou, and V. Zikas, "Incoercible multiparty computation and universally composable receipt-free voting," in *Advances in Cryptology – CRYPTO 2015*, ser. Lecture Notes in Computer Science, vol. 9216. Springer, 2015, pp. 763–780.

[53] T. Haines and B. Smyth, "Surveying definitions of coercion resistance," Cryptology ePrint Archive, Paper 2019/822, 2019. [Online]. Available: https://eprint.iacr.org/2019/822

[54] P. B. Rönne, T. Finogina, and J. Herranz, "Expanding the toolbox: Coercion and vote-selling at vote-casting revisited," in *Electronic Voting – E-Vote-ID 2024*, ser. Lecture Notes in Computer Science, vol. 15014. Springer, 2025, pp. 141–157, 2024 conference proceedings, published in the 2025 book.

[55] V. Cortier, D. Galindo, S. Glondu, and M. Izabachene, "Election verifiability for helios under weaker trust assumptions," in *European Symposium on Research in Computer Security*. Springer, 2014, pp. 327–344.

[56] V. Cortier, D. Galindo, R. Küsters, J. Müller, and T. Truderung, "Sok: Verifiability notions for e-voting protocols," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 779–798.

[57] D. Bernhard, V. Cortier, D. Galindo, O. Pereira, and B. Warinschi, "Sok: A comprehensive analysis of game-based ballot privacy definitions," in *2015 IEEE Symposium on Security and Privacy*. IEEE, 2015, pp. 499–516.

Appendix A. Ethics considerations

This work provides a formal security analysis of existing e-voting protocols and does not involve human subjects, personal voting data, or experiments on deployed election systems. Our analysis identifies the assumptions and adversarial capabilities under which coercion resistance can or cannot be achieved. Although these findings may reveal security limitations that could potentially be exploited by coercers, our results are based on formal security models and do not necessarily imply practical vulnerabilities in deployed systems. To mitigate potential misuse and misinterpretation, we explicitly state the assumptions and limitations of each security claim. Our goal is to help researchers and election administrators assess the security guarantees of e-voting protocols and select appropriate protocols for their intended deployment environments.

Appendix B. Threshold Corruption of Talliers

We extend the corruption-bound axis to express bounds on the number of corrupted talliers, as required by protocols using threshold tallying, such as JCJ [1]. The extension specifies a family of admissible corrupted coalitions, while retaining the corruption mode, coercion phases, time model, and contact budget of Section 3.2.

B.1. Distributed tallier execution

Fix an election descriptor E, a target voter i^*, and a choice m^* as in Definition 6. Let $\mathcal{T} = \{T_1, \ldots, T_n\} \subseteq \mathcal{P}$ be the publicly specified set of distinct tallier parties. When tallying uses a θ-out-of-n decryption scheme, its decryption threshold $1 \leq \theta \leq n$ is also specified by E. Each tallier holds its own current private state td_j, including its key share, when generated, and its retained auxiliary state. The notation

$$\mathsf{td} = (\mathsf{td}_1, \ldots, \mathsf{td}_n)$$

describes the joint state in the execution record; it does not give any party access to the other components.

We retain the eleven external interfaces and allow Setup and Tally to be realized by protocol-specified communicating PPT machines. The descriptor fixes their realization, message-delivery rules, and tally-announcement rule. The experiment executes their actual state transitions, including distributed key generation when prescribed by the protocol. Public messages are released to the adversary, and private messages are delivered according to the specified interfaces. Corrupting a party exposes its retained local state and grants control over all its subsequent computations and messages, including those associated with its other election roles. If $i^* \in \mathcal{T}$, disclosures of its tallier state follow the target's Obey or Res behavior as well. Invalid contributions, withheld messages, rejections, and aborts remain part of the execution.

For a distributed realization, time t indexes the scheduled local-machine activations, starting at 0 before setup, with a fixed polynomial bound L on their total number. An interface implemented by one party has one activation per call. Corruption may occur before each activation, including within distributed setup and tallying. The sets J_B and $J_{B,W}$ are defined on this clock using the realization's public phase and timing information, as in Section 3.2. All capability comparisons use the same realization and clock.

B.2. Admissible corruptions

A *corruption structure* is a publicly specified, nonempty family

$$\mathfrak{F} \subseteq 2^{\mathcal{P} \setminus \{i^*\}}$$

with an efficiently decidable membership predicate, satisfying

$$D \in \mathfrak{F}, \quad D' \subseteq D \quad \Longrightarrow \quad D' \in \mathfrak{F}.$$

Thus $\mathfrak{F}$ is downward closed and contains $\varnothing$. It is fixed before setup, independently of the hidden experiment bit and private execution states. Write

$$U_{\mathfrak{F}} := \bigcup_{D \in \mathfrak{F}} D$$

for the set of parties that occur in an admissible coalition.

For an eligible-party set $C \subseteq \mathcal{P} \setminus \{i^*\}$ and an integer $0 \le f \le n$, the threshold specialization is

$$\mathfrak{F}_{C,\mathcal{T},f} := \{D \subseteq C : |D \cap \mathcal{T}| \le f\}. \tag{3}$$

It permits any subset of the eligible non-tallier parties and any adversarially chosen subset of at most f eligible talliers. In particular, taking $\mathcal{T} \subseteq C$ permits the adversary to choose among all talliers. Every corrupted party in $\mathcal{T}$ counts toward the bound, even if it also performs another role. The parameter f bounds corrupted parties; it is independent of the protocol parameter θ.

Definition 11 (Capabilities with a corruption structure)**.** For a corruption structure $\mathfrak{F}$ and parameters ρ, B, W, k from Definition 7, define

$$\widehat{\Omega}_{\mathfrak{F},\rho,B,W,k} := \big(\Omega_{U_{\mathfrak{F}},\rho,B,W,k}, \mathfrak{F}\big).$$

The experiment maintains the cumulative set D of corrupted parties, initially $D = \varnothing$. In addition to the public eligibility schedule in $\Omega_{U_{\mathfrak{F}},\rho,B,W,k}$, a request to corrupt a new party p at time t must satisfy

$$p \in \mathcal{C}_t^{U_{\mathfrak{F}},\rho} \setminus D, \qquad D \cup \{p\} \in \mathfrak{F}.$$

An accepted request updates $D \leftarrow D \cup \{p\}$ and exposes the party's current local state; a request violating either condition returns $\perp$ without changing D or exposing private state. For $\rho = \mathsf{stat}$, the entire initial coalition is selected before the first setup activation, and no further corruption is permitted. For $\rho = \mathsf{adapt}$, new corruption choices may depend on the adversary's accumulated view, subject to the same membership test after every addition. Corruption is permanent and all previously obtained information is retained. The contact permissions and budget are those of $\Omega_{U_{\mathfrak{F}},\rho,B,W,k}$. An adversary respects $\widehat{\Omega}_{\mathfrak{F},\rho,B,W,k}$ if all its corruption and contact actions satisfy these restrictions.

The public eligibility schedule does not depend on D; the experiment enforces membership in $\mathfrak{F}$ separately. For (3), this enforces $|D \cap \mathcal{T}| \le f$ throughout the execution, including setup and tallying, rather than imposing a separate budget in each phase.

B.3. Coercion resistance and implication relations

Definition 12 (Coercion resistance with threshold corruption)**.** Fix the common input distribution and execution rules, i^*, m^*, and $\delta \in [0,1]$ as in Definition 6. Run the experiment of Figure 2 using the distributed realization above and the corruption rule of Definition 11, with step (8) executing the next scheduled local-machine activation. The target's behavior, the adversary's view, and the goal predicate γ_{i^*,m^*} are otherwise unchanged. In particular, $\gamma_{i^*,m^*} = 0$ if no tally is announced, and no execution is discarded or conditioned on successful completion.

We write

$$\mathsf{CR}^{\mathfrak{F},\rho,B,W,\le k}_{\Pi_{\mathsf{voting}}}(\gamma_{i^*,m^*}, \delta)$$

if there exists a PPT counterstrategy Res such that, for every PPT adversary $\mathcal{A}$ respecting $\widehat{\Omega}_{\mathfrak{F},\rho,B,W,k}$, there is a negligible function η for which

$$\Pr[\gamma_{i^*,m^*}(\tau_b) = 1 \mid b = 1] \ge 1 - \eta(\lambda), \tag{4}$$

$$\left|\Pr[b' = b] - \frac{1}{2}\right| \le \frac{\delta}{2} + \eta(\lambda). \tag{5}$$

Here b is uniform, b' is the adversary's output, and τ_b is the complete execution record, as in the original experiment. The same Res must work for all permitted corruption choices. It may use the public description of $\mathfrak{F}$, public information, and its own local state; the experiment does not additionally disclose D or other parties' private states to it. Threshold corruption uses $\mathfrak{F} = \mathfrak{F}_{C,\mathcal{T},f}$.

Proposition 13 (Recovery and monotonicity)**.** *For a fixed protocol realization and common input and execution rules, taking $\mathfrak{F} = 2^C$ recovers the original corruption policy with corruption bound C. For the basic realization, with one activation per original algorithm call, this recovers the original classified capability and experiment. Moreover, let $\mathfrak{F}_1 \subseteq \mathfrak{F}_2$ and suppose that $\rho_1 = \rho_2$ or $(\rho_1, \rho_2) = (\mathsf{stat}, \mathsf{adapt})$, $J_{B_1,W_1} \subseteq J_{B_2,W_2}$, and $k_1 \le k_2$. Then*

$$\mathsf{CR}^{\mathfrak{F}_2,\rho_2,B_2,W_2,\le k_2}_{\Pi_{\mathsf{voting}}}(\gamma_{i^*,m^*}, \delta)$$
$$\Longrightarrow \mathsf{CR}^{\mathfrak{F}_1,\rho_1,B_1,W_1,\le k_1}_{\Pi_{\mathsf{voting}}}(\gamma_{i^*,m^*}, \delta).$$

In particular, for fixed $C, \mathcal{T}$ and the remaining parameters, security for f_2 corrupted talliers implies security for f_1 corrupted talliers whenever $f_1 \le f_2$.

Proof. If $\mathfrak{F} = 2^C$, then $U_{\mathfrak{F}} = C$ and the additional membership test permits exactly the corruptions allowed originally. Under the stated inclusions, every adversary allowed by the first setting is also allowed by the second, so the latter's counterstrategy satisfies both security conditions in the former with the same bounds. Finally, $f_1 \le f_2$ implies $\mathfrak{F}_{C,\mathcal{T},f_1} \subseteq \mathfrak{F}_{C,\mathcal{T},f_2}$. □

Decryption threshold and tally availability.. The parameter θ belongs to the protocol instance, whereas f specifies an adversarial capability. For ordinary θ-out-of-n key sharing with one share per tallier, when exposure of key shares is confined to corrupted talliers, $f < \theta$ is the usual corruption regime for protecting the shared key, while $n - f \ge \theta$ ensures that at least θ talliers remain uncorrupted. These numerical conditions alone do not establish coercion resistance: the distributed protocol must provide the required secrecy and completion properties under the selected corruption mode and execution rules. In particular, suppose every tally announcement requires contributions from at least θ distinct talliers, with no alternative announcement mechanism. For $\mathfrak{F}_{C,\mathcal{T},f}$ with $\mathcal{T} \subseteq C$, if $n - f < \theta$, the adversary can corrupt $T_1, \ldots, T_f$ before setup and withhold all their contributions. This prevents a tally from being announced and violates (4) for every counterstrategy. Thus threshold parameters are fixed when comparing capabilities, and each protocol's achieved corruption bound must account

for both information exposure and its tally-announcement requirements.