Exploring boundaries of privacy in e2e verifiable e-voting systems.

Tamara Finogina

1 Preliminaries

1.1 Notations

Through this paper, we denote λ as the security parameter. We use $negl(\lambda)$ to denote a negligible function in λ, i.e., it always holds that $negl(\lambda) < \frac{1}{\lambda^c}$ for any $0 < c \in \mathbb{Z}$ for sufficient large λ.

2 E-voting model

2.1 Syntax

An e-voting system is an interactive protocol Π among a set of entities and voters, that takes voters' preferences as and input and aims to return a tally.

In this modelling, the election system involves five types of entities, the voters $V_1, \ldots, V_n$, possibly equipped with the voting supporting device (VSD) and the auditing supporting device (ASD), the election authority (EA), the vote collector (VC), the trustee (T), and the bulletin board (BB) whose only role is to provide storage for the election transcript for the purpose of verification.

(1) BB is completely passive and is only writeable by the VC and readable by anyone.

(2) Voters submit their votes by engaging in the ballot casting protocol to the VC and they are not allowed to interact with each other.

(3) VC only role is to collect votes and write them to BB

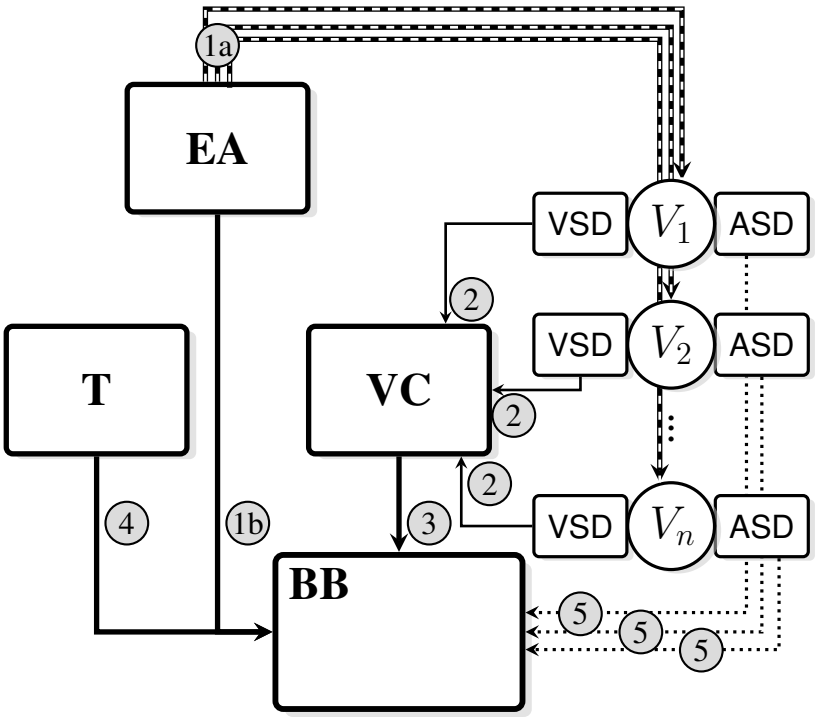


FIGURE 1: The interaction among the entities in an e-voting execution. The dotted lines denote read-only access to the BB. The dashed arrows denote channels for voters' private inputs distribution. Annotation: (1a): distribution of voter's private inputs; (1b): pre-election BB data; (2): vote casting; (3): writing votes to BB; (4) post-election BB data; (5): auditing.

(4) T is responsible for computing the tally and announcing the election result.

(5) EA - prepares all the election setup information and distributes the voters' ballots.

Protocols An e-voting system Π is a quintuple of algorithms and protocols **(Setup, Cast, Tally, Result, Verify)** specified as follows:

(1) The algorithm **Setup** is executed by the EA and T separately. During the setup phase EA generates Π's public parameters Pub (which include P, V, U) and the voters' secrets $s_1, \ldots, s_n$. The part of the algorithm during which EA distributes secrets among voters is defined as **Registration**. At the same time, T generates pre-election BB data and posts it on BB.

(2) The interactive protocol **Cast** is executed between three parties, the voter V_l, the BB and the VC. During this interaction, the voter uses VSD, his secret s_l and an option U_l to generate the ballot b_l and sends this ballot to VC. Upon successful termination, VC posts ballot b_l to BB and the voter V_l receives a receipt α_l.

(3) The algorithm **Result** is executed by T and outputs the result τ for the election or returns $\perp$ in case such result is undefined.

(4) The algorithm **Verify** outputs a value in $\{0, 1\}$, where α is a voter receipt (that corresponds to the voter's output from the **Cast** protocol).

In some e-voting systems **Registration** part is omitted and voters are expected to receive their credentials via secure channel such as post mail, pulling place etc.

2.2 Correctness of a system

We say that a system Π has (perfect) correctness, if for any honest execution of any subset of not abstained voters that results in a public transcript τ, where the voters $V_1, ..., V_n$ cast votes for options $U_1, ..., U_n$, it holds that $Result(\tau) = f(U_1, ..., U_n)$, where $f(U_1, ..., U_n)$ is the m-vector whose i-th location is equal to the number of times a candidate $P_i \in \{P_1, \ldots, P_m\}$ was chosen in the candidate selections $U_1, ..., U_n$.

3 Privacy with respect to entities

3.1 EA only

If EA is trusted and all other entities $(T, \mathrm{VSD}, \mathrm{VC})$ are corrupted (BB is completely passive and simply posts all information received from VC) it is impossible to preserve voter's privacy.

Proof:

Suppose there is some e-voting system Π which doesn't allow any adversary to learn anything about individual voters' intents in case when only EA is trusted. $\mathcal{A}$ strategy is to calculate tally after every vote submission and compare it with the previous tally to learn the voter's intent. Due to a correctness, Π should be

able to produce meaningful election tally for any turnout, including just one voter. Since an adversary $\mathcal{A}$ controls a trustee T, he should be able to compute results after very first cast procedure is done. This means that either $\mathcal{A}$ may learn voters' choices one by one or Π fails to produce results for any subset of abstained voters and therefore is not correct.

3.1.1 EA + VC

It's not enough for privacy, an adversary can perform exactly the same attack as for 'EA only case'

3.1.2 EA + T

Private. Example: Demos

3.1.3 EA + VSD

In such case Π can be private, since it's enough to have a trusted VSD for preserving voters' privacy.

3.2 VSD only:

VSD is trusted, all other entities (T, EA, VC) are controlled by an adversary. BB is completely passive and simply posts all information received from VC. Consider the following e-voting system Π:

(1) EA generates a large random number R and commitments bases g, h. Also EA picks a random id x_i for every candidate $cand_i$. $R, g, h, \{cand_i, x_i\}$ - Π's public information.

(2) each voter submits vis VSD a perfectly hiding commitment $g^{x_i}h^{r_l}$, where r_l is a random number less than R picked by VSD.

(3) after the last voter cast his vote, VSD submits a commitment $g^{x_{abs}}h^{r_{vsd}}$, where x_{abs} corresponds to the abstain option and $r_{vsd} = R - \sum_{l=0}^{l=n} r_l$, n - number of voters.

(4) T computes the product of all hiding commitments and opens the result.

In order to break voters' privacy, an adversary has to break perfectly hiding commitments.

3.3 VC only

Impossible. An adversary can perform exactly the same attack as for 'EA only case.

3.3.1 VC + T

Due to the assumption that voters are not able to collaborate for protecting their privacy and all complex math operation performed by VSD or EA, it's impossible to have privacy in this case. A voter provides VSD with an input which is either in a plain text or calculated by EA line and therefore immediately gives away his choice.

3.3.2 VC + VSD

In such case Π can be private, since it's enough to have a trusted VSD for preserving voters' privacy.

3.3.3 VC + EA

Impossible. See the EA + VC case for details.

3.4 T only

Impossible. Same reasoning as VC + T case.

3.4.1 T + VC

seems meaningless

3.4.2 T + VSD

In such case Π can be private, since it's enough to have a trusted VSD for preserving voters' privacy.

3.4.3 T + EA

Private. See the EA + T case for details.

4 Strict privacy notion

Voter privacy implies that a voter is capable of casting his own vote secretly and freely without letting others' parties, namely an adversary, to learn some information about his preferences or interfere in it.

In general, the goal of the adversary who attacks voter privacy is to learn some information about the candidate selections of the honest voters. We define an attack against voters privacy successful, if there is an election result*, for which an adversary is capable of distinguishing how the honest voters voted while it can observe the whole e-voting network** and corrupt some part of honest voter and some entities and also has access to honest voters' receipts.

*Obviously, it doesn't include trivial election results, where all voters voted for the same option.
**Except channels that are marked as untrappable.

In this section we use term strict privacy to imply that an adversary interacts with an e-voting system on behalf of all voters and there is no such thing as honest voters. In general, strict privacy means that even the person who cast a vote can not break his own privacy after the Cast phase is completed.

We formally define the strict voter privacy via a Voter Privacy game, denotes as $G^{\mathcal{A},Sim}_{strict,<honest\ entities>}(1^{\lambda})$, that is played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, that takes as input the security parameter λ* and returns 1 or 0 depending on whether the adversary wins. Also, $\mathcal{A}$ is allowed to corrupt some entities. The choice of the corrupted parties splits the Voter Privacy game into two different scenarios. All meaningful* cases of collusion fall into two scenarios: (1) entities (EA, T) are hones and (2) VSD is honest.
Thomas, do you think that m and n (the number of candidates and the number of voters) should be added as the input parameters to the Voter Privacy game? Now, it's up to adversary to define an election parameters.
*In cases where only EA or T can be trusted it is not possible to have voter privacy.

According to the game rules, an adversary is allowed to define an election parameters, act on behalf of all voters and corrupted entities. During the game $\mathcal{C}$ plays the role of honest parties and returns to the adversary simulated and real view in order defined by a coin a (If $a = 0$, $\mathcal{C}$ returns $(simulated\_view, real\_view)$

and $(real\_view, simulated\_view)$ otherwise). A challenger flips the coin a only once, before interaction with $\mathcal{A}$. Note that the result of the game heavily depends on an efficiency of the simulator that is used by the challenger for producing a simulated view.

We will show that the notion of strict privacy is the weakest level of privacy that contradicts end-to-end verifiability.

4.1 EA and T are honest: $G^{\mathcal{A},Sim}_{strict,\mathrm{EA},T}(1^\lambda)$

In the case of trusted EA and T, $\mathcal{C}$ uses a simulator Sim to generate a fake credentials so an adversary would not be able to distinguish the result of the **Cast** protocol with his intent and real credentials and with another candidates selection and fake credential given as input.

$\mathcal{C}$ behaviour in the $G^{\mathcal{A},Sim}_{strict,\mathrm{EA},T}(1^\lambda)$ game captures the ability of a honest voter to lie about his vote in code-based e-voting schemes. Suppose, an adversary makes a voter V to cast his vote for an option $U_\mathcal{A}$ instead of the voter's original intent U_V. V still can cast vote for the option U_V and fake his credentials in a way that his actual receipt would correspond to the option $U_\mathcal{A}$. If by faking credential V can fool an adversary into believing that he cast vote for $U_\mathcal{A}$, his privacy is preserved.

For every vote that an adversary cast, $\mathcal{C}$ creates a fake voter who cast exactly the same vote. Every round the sum of an adversarial vote and an challenger's vote would give $U_\mathcal{A}$ + U_V. An actual vote distribution depends on the credentials choice. If $\mathcal{A}$ uses real credentials he votes for the $U_\mathcal{A}$ and $\mathcal{C}$ – for the U_V. Otherwise , $\mathcal{A}$' vote corresponds to the option U_V and $\mathcal{C}'$ vote to the $U_\mathcal{A}$. $\mathcal{A}$ is provided with both credentials real and fake in order defined by $\mathcal{C}$'s coin a. If $a = 0$ $\mathcal{C}$ returns $(fake\_credentials, real\_credentials)$ and $(real\_credentials, fake\_credentials)$ otherwise. An adversary tries to guess the coin a. $\mathcal{A}$ may stop the game in any moment *.
Not really. An adversary can not stop the game in the moment after he cast his vote and before $\mathcal{C}$ posts a fake voter's vote.

In the game $G^{\mathcal{A},Sim}_{strict,\mathrm{EA},T}(1^\lambda)$, an adversary $\mathcal{A}$ interacts with the challenger $\mathcal{C}$ on behalf of all voters, VC and VSD. $\mathcal{C}$ plays the role of EA and T. BB is completely passive and represents a publicly viewed database.

After an adversary $\mathcal{A}$ defined an election parameters, $\mathcal{C}$ doubles the voters, com-

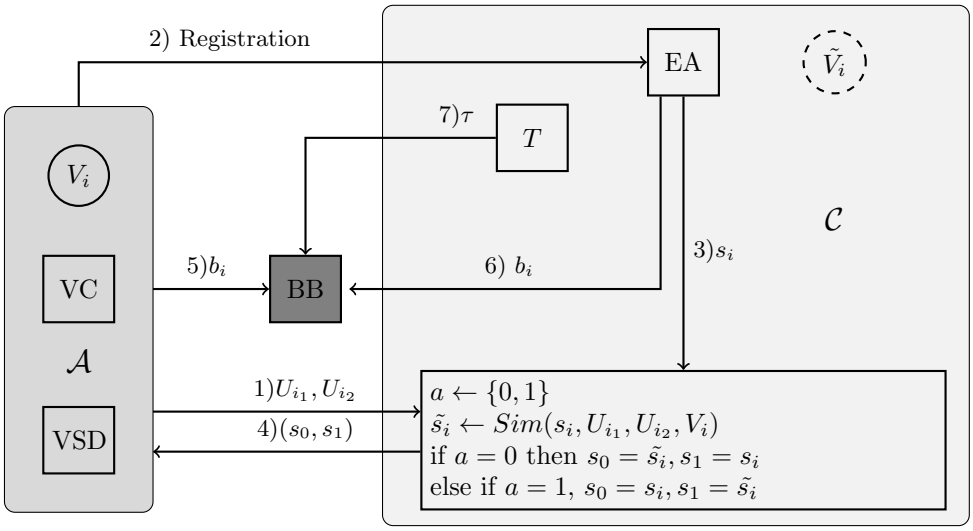


FIGURE 2: $G^{\mathcal{A},Sim}_{strict,\mathrm{EA},T}(1^\lambda)$

putes setup data and starts the election. This duplication of voters is needed because $\mathcal{C}$ would post another ballot for every adversarial's cast ballot. If $\mathcal{A}$ abstains, $\mathcal{C}$ abstains as well. During the game, we will refer to duplicated voters as fake ones.

$\mathcal{A}$ 1) picks and sends two options U_{i_1}, U_{i_2} * to the challenger $\mathcal{C}$. One of the options U_{i_1}, U_{i_2} is his intent, the other – the option that the challenger would use in order to produce an indistinguishable from the intent's ballot and receipt view*. After sending options, 2) $\mathcal{A}$ runs the **Registration** as if he is V_i. $\mathcal{C}$ creates a fake credentials $\tilde{s_i}$ and 3) generates real credentials s_i for him. 4) $\mathcal{C}$ responds $\mathcal{A}$ with a pair of credentials s_0, s_1, where one of the credentials are real and the other were generated using the simulator Sim in a such way, that no matter what credential s_0 or s_1 $\mathcal{A}$ uses to cast a vote for U_{i_1}, the produced ballot would also correspond to the result of the **Cast** protocol for an option U_{i_2} with the other credentials and vice versa. If 5) $\mathcal{A}$ choses to post the ballot b_i to BB, 6) $\mathcal{C}$ posts exactly the same ballot to BB on behalf of the fake voter. When $\mathcal{A}$ stops the election, 7) $\mathcal{C}$ posts the tally τ **.

*Haven't thought about the case when $U_{i_1} = U_{i_2}$. Should the challenger return same credentials or Sim would generate something different? On the one hand,

it's fine for a Sim to generate exactly the same credentials. If it generates something different, the result of the game depends on how good Sim is. On the other hand, case $U_{i_1} = U_{i_2}$ is a bit weird. This game tries to capture a way how a honest voter fakes his credentials to convince $\mathcal{A}$ that he voted as he was told. There is no point in faking anything if you may as well give back real stuff.

Remarks:

*If $\mathcal{C}$ succeeds, $\mathcal{A}$ wouldn't be able to say whether his ballot and receipt corresponds to U_{i_1} or U_{i_2} and BB would contain both ballots (one for the option U_{i_1} ,the other for the option U_{i_2}) so the tally wouldn't reveal any information. Example: if $\mathcal{A}$ picks real credential for the voter V_i then he casts a vote for the option U_{i_1} , at the same time exactly the same ballot posted $\mathcal{C}$ on behalf of the fake voter $\tilde{V}_i$ would correspond to the fake credentials and the option U_{i_2}. In this case $\mathcal{A}$ voted for U_{i_1}. Else if $\mathcal{A}$ picks fake credentials and tries to vote for the option U_{i_1}, his ballot corresponds to the real credentials and the option U_{i_1}. At the same time, exactly the same ballot posted $\mathcal{C}$ on behalf of the fake voter $\tilde{V}_i$ would correspond to the fake credentials and the option U_{i_1}. So $\mathcal{A}$ voted for U_{i_1}. If this definition holds, $\mathcal{A}$ has no idea whom he voted for.

** The tally τ is posted only if all ballots are correctly formed: were produced with one of the provided credentials, for the candidate U_{i_1}, for every cast ballot there is a duplicate, etc.

Thomas, may be you are right and here I should use two sets of votes that sums to the same tally. I'm not entirely sure, though. This approach seems quite similar to the 'only VSD is trusted' case. Here $\mathcal{A}$ can't say wether he voted for U_{i_1} and $\mathcal{C}$ for U_{i_2} or vice versa. On the one hand, he knows options U_{i_1} and U_{i_2} and just couldn't link them with voters. U_{i_1} and U_{i_2} are defined by $\mathcal{A}$ and keep changing during the game. Generally, in order to save my privacy, I should be able to lie about my intent. Which means that if adversary told me to vote for U_{i_1} and I voted for U_{i_2} instead, he should not detect the lie. He even gets less information, since I definitely not going to give him 2 views. This definition addresses exactly this type of lying, however it's not how you defined privacy in Demos. Here $\mathcal{A}$ knows that both options in BB and knows which entry corresponds to v_i and which one to $\tilde{V}_i$, in Demos only one ballot is posted. Is lying about just one pair is enough? $\mathcal{A}$ himself is not sure wether he voted for U_{i_1} or for U_{i_2} (if he picks the real credentials then it's U_{i_1}, otherwise – U_{i_2}). Here, I've tried to give an adversary as many information and freedom to operate as possible: picking options adaptively instead of fixed set and receiving 2 views instead of just a single one. Does anything here make sense?

$G^{\mathcal{A},Sim}_{strict,\mathrm{EA},T}(1^\lambda)$ defined as follows:

(1) During the game $\mathcal{C}$ plays the role of EA and T. $\mathcal{A}$ operates on behalf of the all voters, VSD and VC.

(2) $\mathcal{A}$ defines a set of voters $\mathcal{V} = \{V_1, ..., V_n\}$, a list of candidates $\mathcal{P} = \{P_1, ..., P_m\}$, a set of allowed candidates' selections $\mathcal{U}$. It provides $\mathcal{C}$ with $\mathcal{V}, \mathcal{P}, \mathcal{U}$.

(3) $\mathcal{C}$ doubles the set $\mathcal{V}$ adding fake voters $\{V'_1, ..., V'_n\}$ and starts the election on behalf of EA. Also, $\mathcal{C}$ flips a coin $a \leftarrow \{0, 1\}$ to define an order according to which real and simulated credentials would be returned to $\mathcal{A}$.

(4) The adversary $\mathcal{A}$ picks two option $U_{i_1}, U_{i_2} \in \mathcal{U}$, where U_{i_1} is an option for the real credentials and U_{i_2} is an option for the fake ones. After that, $\mathcal{A}$ and $\mathcal{C}$ engage in an interaction where $\mathcal{A}$ runs the **Registration** protocols on behalf of all voters. For each voter $V_i \in \mathcal{V}$:

– $\mathcal{C}$ picks a fake voter $\tilde{V}_i$ and generates credentials $\tilde{s}_i$ for him using Sim*. $\mathcal{C}$ responds $\mathcal{A}$ with a pair of simulated and real credentials (s_0, s_1) in order defined by the coin a:

$$\begin{cases} \text{if } \ a = 0, \ \ (s_0, s_1) = (\tilde{s}_i, s_i) \\ \text{else } \ (s_0, s_1) = (s_i, \tilde{s}_i) \end{cases}$$

– Using one of the credentials $\mathcal{A}$ schedules the **Cast** protocol executions and sends the produced ballot to $\mathcal{C}$.

– If $\mathcal{A}$ posts a ballot to BB, $\mathcal{C}$ posts exactly the same ballot in the entry that corresponds to the fake voter $\tilde{V}_i$

(5) $\mathcal{C}$ executes the $Tally$ protocol.

(6) Finally, $\mathcal{A}$ using all information collected above (including the contents of the BB) outputs a bit a^*

(7) The game returns a bit which is 1 if $a = a^*$ and 0 otherwise.

***Remark:**
Sim works in a way that the fake credentials satisfy both of the following rules:

(1) The real credentials and U_{i_1} option should give ballot and receipt, which are identical* to ballot and receipt produced for the fake credentials and U_{i_2} option.

(2) The fake credentials and U_{i_1} option should give ballot and receipt, which are identical to ballot and receipt produced for the real credentials and U_{i_2} option.

* indistinguishable ?

Strict privacy: EA **and** T **are honest:**

The e-voting system Π achieves strict voter privacy in case of trusted EA and T, if there is a PPT simulator Sim such that for any PPT adversary $\mathcal{A}$:

$$|\Pr[G^{\mathcal{A},Sim}_{strict,\mathrm{EA},T}(1^\lambda) = 1] - \tfrac{1}{2}| = negl(\lambda)$$

4.2 VSD is honest: $G^{\mathcal{A},Sim}_{strict,\mathrm{VSD}}(1^\lambda)$

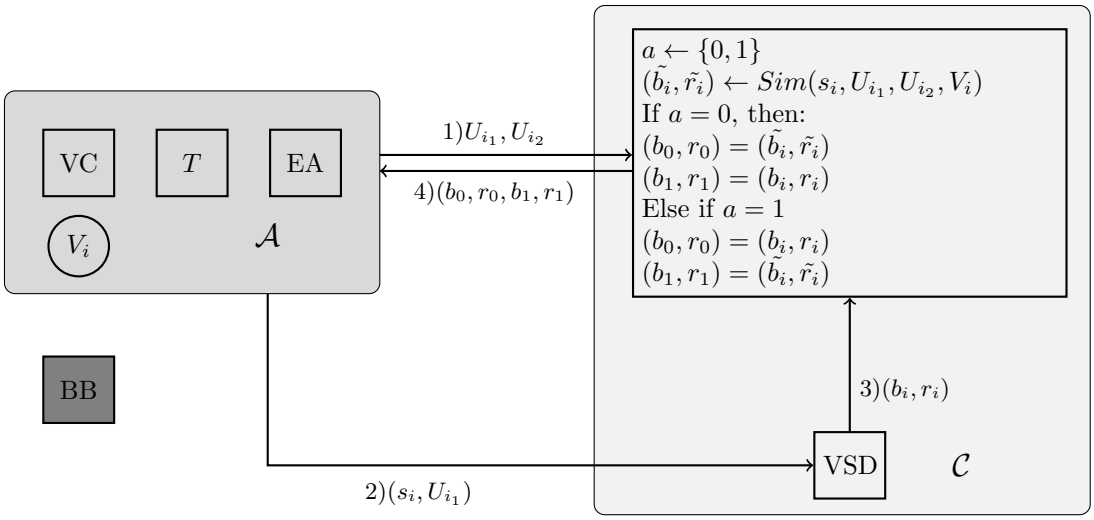


FIGURE 3: $G^{\mathcal{A},Sim}_{strict,\mathrm{VSD}}(1^\lambda)$

In the game defined above, $\mathcal{A}$ operates on behalf of the all voters and all corrupted entities, such as: EA, VC and T. BB is completely passive and represents a publicly accessible database.
1) an adversary $\mathcal{A}$ picks and sends options U_{i_1}, U_{i_2} to the challenger $\mathcal{C}$. After that 2) $\mathcal{A}$ schedules the **Cast** protocol with $\mathcal{C}$ as if he is a voter V_i. $\mathcal{C}$ 3) generates a real

ballot and receipt and uses Sim to create a fake ones. At the end 4) $\mathcal{C}$ responses with a pair of ballots and receipts b_0, r_0, b_1, r_1, where one ballot and receipt corresponds to an option U_{i_1}, the other was generated using the simulator Sim for an option U_{i_2}.

The game $G^{\mathcal{A},Sim}_{strict,\mathrm{VSD}}(1^\lambda)$ is defined as follows:

(1) During the game $\mathcal{C}$ plays the role of VSD. $\mathcal{A}$ operates on behalf of the all voters and all corrupted entities, such as: EA, VC and T.

(2) $\mathcal{A}$ defines a set of voters $\mathcal{V} = \{V_1, ..., V_n\}$, a list of candidates $\mathcal{P} = \{P_1, ..., P_m\}$, a set of allowed candidates' selections $\mathcal{U}$ and starts the election.

(3) $\mathcal{C}$ flips a coin $a \leftarrow \{0, 1\}$ to define an order according to which real and simulated ballots and receipts would be returned to $\mathcal{A}$.

(4) The adversary A and the challenger $\mathcal{C}$ engages in an interaction where $\mathcal{A}$ runs the Cast protocols on behalf of all voters. For each voter $V_i \in \mathcal{V}$:

– $\mathcal{A}$ sends to $\mathcal{C}$ options U_{i_1}, U_{i_2} and starts with $\mathcal{C}$ the **Cast** protocol on behalf of V_i. As a result of the protocol execution $\mathcal{C}$ provides $\mathcal{A}$ with a pair of simulated and real ballot and receipt $(b_0, r_0)(b_1, r_1)$ s.t.:
$$\begin{cases} \text{if } \ a = 0, \ \ (b_0, r_0) = (\tilde{b_i}, \tilde{r_i}) \ \text{ and } \ (b_1, r_1) = (b_i, r_i) \\ \text{else } \ (b_0, r_0) = (b_i, r_i) \ \text{ and } \ (b_1, r_1) = (\tilde{b_i}, \tilde{r_i}) \end{cases}$$
where the pair (b_i, r_i) is the ballot and receipt for an adversarial option U_{i_1} and $(\tilde{b_i}, \tilde{r_i})$ is the ballot and receipt for U_{i_2} option generated via the simulator Sim.

(5) Finally, $\mathcal{A}$ executes the $Tally$ protocol and using all information collected above outputs a bit a^*

(6) The game returns a bit which is 1 if $a = a^*$ and 0 otherwise

Strict privacy: VSD **is honest:**

The e-voting system Π achieves strict voter privacy in case when VSD is trusted, if there is a PPT simulator Sim such that for any PPT adversary $\mathcal{A}$:

$$|\Pr[G^{\mathcal{A},Sim}_{strict,\mathrm{VSD}}(1^\lambda) = 1] - \tfrac{1}{2}| = negl(\lambda)$$

5 Privacy

We formally define the voter privacy via a Voter Privacy game, denotes as $G^{\mathcal{A},Sim}_{t-priv,<honest\ entities>}(1^\lambda)$, that is played between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$, that takes as input the security parameter λ and returns 1 or 0 depending on whether the adversary wins. Also, $\mathcal{A}$ is allowed to corrupt some entities. The choice of the corrupted parties splits the Voter Privacy game into two different scenarios: (1) entities (EA, T) are hones and (2) VSD is honest.
The only difference between Strict privacy and Privacy is the number of corrupted voters. In Strict privacy games **all** voters are corrupted and $\mathcal{A}$ vote on their behalf. On the contrary, in the Privacy games at most t voters are corrupted and all others are honest. The crucial moment is that honest voters are able to verify that their vote was cast as intended, but an adversary can not since he does not know the intent and does not control all entities. Therefore, voters may lie about vote while an adversary have no ways to learn the truth and not to win the game $G^{\mathcal{A},Sim}_{t-priv,<honest\ entities>}(1^\lambda)$.

5.1 EA and T are honest: $G^{\mathcal{A},Sim}_{t-priv,\mathrm{EA},T}(1^\lambda)$

In the game $G^{\mathcal{A},Sim}_{t-priv,\mathrm{EA},T}(1^\lambda)$, an adversary $\mathcal{A}$ interacts with the challenger $\mathcal{C}$ on behalf of all corrupted voters, VC and VSD. $\mathcal{C}$ plays the role of honest voters, EA and T. BB is completely passive and represents a publicly viewed database.

After an adversary $\mathcal{A}$ defined an election parameters, $\mathcal{C}$ doubles the voters, computes setup data and starts the election. This duplication of voters is needed because $\mathcal{C}$ would post another ballot for every adversarial's cast ballot. If $\mathcal{A}$ abstains, $\mathcal{C}$ abstains as well. During the game, we will refer to duplicated voters as fake ones.

$\mathcal{A}$ 1) picks and sends two options U_{i_1}, U_{i_2} to the challenger $\mathcal{C}$. One of the options U_{i_1}, U_{i_2} is his intent, the other – the option that the challenger would use in order to produce an indistinguishable from the intent's ballot and receipt view*. After sending options, 2) $\mathcal{A}$ schedules the **Registration** protocol with $\mathcal{C}$ on behalf

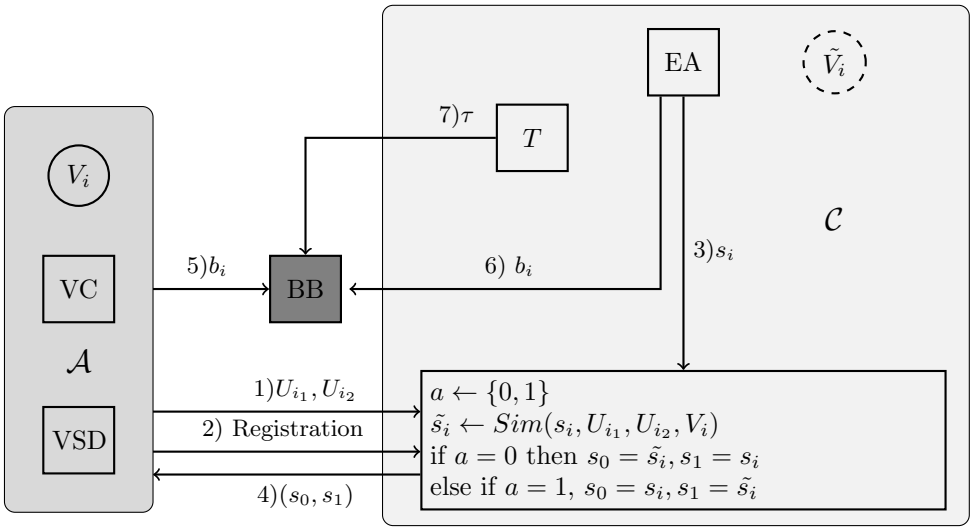


FIGURE 4: $G^{\mathcal{A},Sim}_{t-priv,\mathrm{EA},T}(1^\lambda)$

of some voter V_i. $\mathcal{C}$ creates a fake credentials $\tilde{s_i}$ and 3) generates real credentials s_i for him. 4) $\mathcal{C}$ responds $\mathcal{A}$ with a pair of credentials s_0, s_1, where one of the credentials are real and the other were generated using the simulator Sim in a such way, that no matter what credential s_0 or s_1 $\mathcal{A}$ uses to cast a vote for U_{i_1}, the produced ballot would also correspond to the result of the **Cast** protocol for an option U_{i_2} with the other credentials and vice versa. If 5) $\mathcal{A}$ choses to post the ballot b_i to BB, 6) $\mathcal{C}$ posts exactly the same ballot to BB on behalf of the fake voter. When $\mathcal{A}$ stops the election, 7) $\mathcal{C}$ posts the tally τ **.

Remarks:
*If $\mathcal{C}$ succeeds, $\mathcal{A}$ wouldn't be able to say whether his ballot and receipt corresponds to U_{i_1} or U_{i_2} and BB would contain both ballots (one for the option U_{i_1} ,the other for the option U_{i_2}) so the tally wouldn't reveal any information. Example: if $\mathcal{A}$ picks real credential for the voter V_i then he casts a vote for the option U_{i_1} , at the same time exactly the same ballot posted $\mathcal{C}$ on behalf of the fake voter $\tilde{V_i}$ would correspond to the fake credentials and the option U_{i_2}. In this case $\mathcal{A}$ voted for U_{i_1}. Else if $\mathcal{A}$ picks fake credentials and tries to vote for the option U_{i_1}, his ballot corresponds to the real credentials and the option U_{i_1}. At the same time, exactly the same ballot posted $\mathcal{C}$ on behalf of the fake voter $\tilde{V_i}$ would correspond to the fake credentials and the option U_{i_1}. So $\mathcal{A}$ voted for U_{i_1}. If this definition

holds, $\mathcal{A}$ has no idea whom he voted for.
** The tally τ is posted only if all ballots are correctly formed: were produced with one of the provided credentials, for the candidate U_{i_1}, for every cast ballot there is a duplicate, etc.

$G^{\mathcal{A},Sim}_{t-priv,\mathrm{EA},T}(1^\lambda)$ defined as follows:

(1) During the game $\mathcal{C}$ plays the role of honest voters and EA. $\mathcal{A}$ operates on behalf of corrupted voters and VSD.

(2) $\mathcal{A}$ defines a set of voters $\mathcal{V} = \{V_1, ..., V_n\}$, a list of candidates $\mathcal{P} = \{P_1, ..., P_m\}$, a set of allowed candidates' selections $\mathcal{U}$. It provides $\mathcal{C}$ with $\mathcal{V}, \mathcal{P}, \mathcal{U}$.

(3) $\mathcal{C}$ doubles the set $\mathcal{V}$ adding fake voters $\{V'_1, ..., V'_n\}$ and starts the election on behalf of EA. Also, $\mathcal{C}$ flips a coin $a \leftarrow \{0, 1\}$ to define an order according to which real and simulated credentials would be returned to $\mathcal{A}$.

(4) The adversary $\mathcal{A}$ picks two option $U_{i_1}, U_{i_2} \in \mathcal{U}$, where U_{i_1} is an option for the real credentials and U_{i_2} is an option for the fake ones. After that, $\mathcal{A}$ and $\mathcal{C}$ engage in an interaction where $\mathcal{A}$ schedules the $Registration$ protocols, during which all voters receive their credentials and forward them to $\mathcal{A}$. For each voter $V_i \in \mathcal{V}$, the adversary chooses whether V_i is corrupted:

– If V_i is corrupted, then $\mathcal{C}$ provides $\mathcal{A}$ with the real credentials s_i, and then they engage in a $Cast$ protocol where A vote on behalf of V_i and $\mathcal{C}$ plays the role of EA.

– If V_i is not corrupted, then $\mathcal{C}$ generates real credentials s_i. Also $\mathcal{C}$ picks a fake voter $\tilde{V}_i$ and generates credentials $\tilde{s}_i$ for him using Sim*. $\mathcal{C}$ responds $\mathcal{A}$ with a pair of simulated and real credentials (s_0, s_1) in order defined by the coin a: $\begin{cases} \text{if } \ a = 0, \ \ (s_0, s_1) = (\tilde{s}_i, s_i) \\ \text{else } \ (s_0, s_1) = (s_i, \tilde{s}_i) \end{cases}$

– Honest voters forward both credentials to $\mathcal{A}$

– Using one of the credentials $\mathcal{A}$ schedules the $Cast$ protocol executions and sends the produced ballot to BB.

– If $\mathcal{A}$ posts a ballot on BB, $\mathcal{C}$ posts exactly the same ballot in the entry that corresponds to the fake voter $\tilde{V}_i$

(5) $\mathcal{C}$ executes the $Tally$ protocol.

(6) Finally, $\mathcal{A}$ using all information collected above (including the contents of the BB) outputs a bit a^*

(7) Denote the set of corrupted voters as $\mathcal{V}_{corr}$ and the set of honest voters as $\tilde{\mathcal{V}} = \mathcal{V} \backslash \mathcal{V}_{corr}$. The game returns a bit which is 1 if and only if the following hold true:

   (a) $a = a^*$

   (b) $|\mathcal{V}_{corr}| \leq t$ (i.e., the number of corrupted voters is bounded by t).

***Remark:**
Sim works in a way that the fake credentials satisfy both of the following rules:

(1) The real credentials and U_{i_1} option should give ballot and receipt, which are identical* to ballot and receipt produced for the fake credentials and U_{i_2} option.

(2) The fake credentials and U_{i_1} option should give ballot and receipt, which are identical to ballot and receipt produced for the real credentials and U_{i_2} option.

Privacy: EA **and** T **are honest:**

The e-voting system Π achieves voter privacy in case of honest T and EA for at most t corrupted voters if there is a PPT simulator Sim such that for any PPT adversary $\mathcal{A}$:

$$|\Pr[G^{\mathcal{A},Sim}_{t-priv,\mathrm{EA},T}(1^\lambda) = 1] - \tfrac{1}{2}| = negl(\lambda)$$

5.2 VSD is honest: $G^{\mathcal{A},Sim}_{t-priv,\mathrm{VSD}}(1^\lambda)$

In the game $G^{\mathcal{A},Sim}_{t-priv,\mathrm{EA},T}(1^\lambda)$, an adversary $\mathcal{A}$ operates on behalf of the all voters and all corrupted entities, such as: EA, VC and T. BB is completely passive and represents a publicly accessible database.

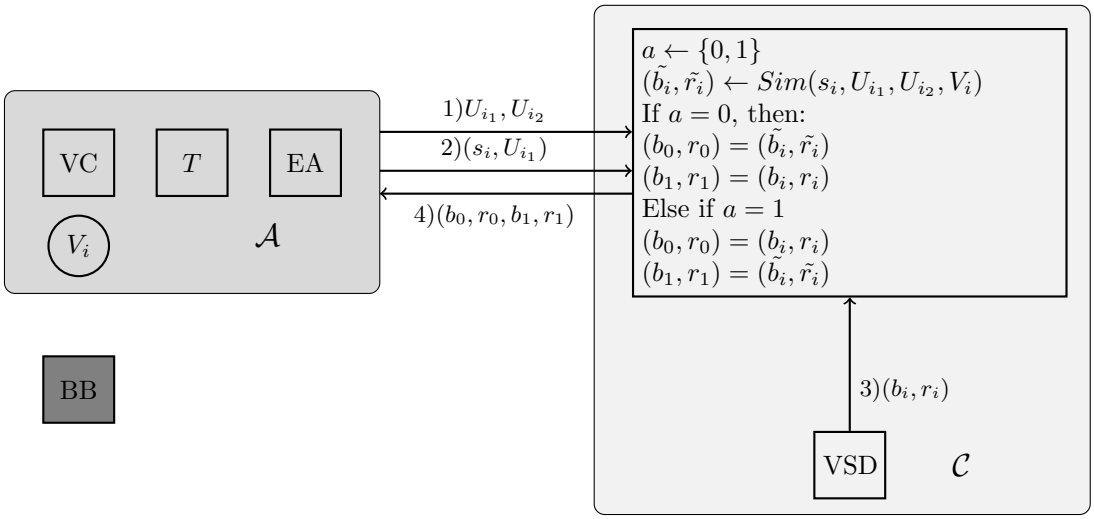


FIGURE 5: $G^{\mathcal{A},Sim}_{t-priv,\mathrm{VSD}}(1^\lambda)$

1) an adversary $\mathcal{A}$ picks and sends options U_{i_1}, U_{i_2} to the challenger $\mathcal{C}$. After that 2) $\mathcal{A}$ schedules the **Cast** protocol with $\mathcal{C}$ on behalf of some voter V_i*. $\mathcal{C}$ 3) generates a real ballot and receipt and uses Sim to create a fake ones. At the end 4) $\mathcal{C}$ responses with a pair of ballots and receipts b_0, r_0, b_1, r_1, where one ballot and receipt corresponds to an option U_{i_1}, the other was generated using the simulator Sim for an option U_{i_2}.

*Remark:
Existence of honest voters is the only difference from the Strict privacy definition. In the strict privacy game, all voters are corrupted.

The game $G^{\mathcal{A},Sim}_{t-priv,\mathrm{VSD}}(1^\lambda)$ is defined as follows:

(1) During the game $\mathcal{C}$ plays the role of the honest voters and VSD. $\mathcal{A}$ operates on behalf of corrupted voters and may corrupt T, and EA.

(2) $\mathcal{A}$ defines a set of voters $\mathcal{V} = \{V_1, ..., V_n\}$, a list of candidates $\mathcal{P} = \{P_1, ..., P_m\}$, a set of allowed candidates' selections $\mathcal{U}$ and starts the election.

(3) $\mathcal{C}$ flips a coin $a \leftarrow \{0,1\}$ to define an order according to which real and simulated ballots and receipts would be returned to $\mathcal{A}$.

(4) $\mathcal{A}$ sends to $\mathcal{C}$ options $U_{i_1}, U_{i_2} \in \mathcal{U}$, where U_{i_1} is an option for the real ballot

and receipt and U_{i_2} is an option for the fake ones. After that, $\mathcal{A}$ and $\mathcal{C}$ engage in an interaction where $\mathcal{A}$ schedules the **Cast** protocols of all voters which may run concurrently. For each voter $V_i \in \mathcal{V}$, the adversary chooses whether $V_i \in \mathcal{V}$ is corrupted:

– If V_i is corrupted, then $\mathcal{C}$ provides $\mathcal{A}$ with the real ballot and receipt (b_i, r_i).

– If V_i is not corrupted, $\mathcal{C}$ provides $\mathcal{A}$ with provides $\mathcal{A}$ with a pair of simulated and real ballot and receipt $(b_0, r_0)(b_1, r_1)$ s.t.:
$\begin{cases} \text{if } \ a = 0, \ \ (b_0, r_0) = (\tilde{b_i}, \tilde{r_i}) \ \text{ and } \ (b_1, r_1) = (b_i, r_i) \\ \text{else } \ (b_0, r_0) = (b_i, r_i) \ \text{ and } \ (b_1, r_1) = (\tilde{b_i}, \tilde{r_i}) \end{cases}$
where the pair (b_i, r_i) is the ballot and receipt for an adversarial option U_{i_1} and $(\tilde{b_i}, \tilde{r_i})$ is the ballot and receipt for U_{i_2} option generated via the simulator Sim.

(5) Finally, (if T is corrupted) $\mathcal{A}$ executes the $Tally$ protocol and using all information collected above outputs a bit a^*

(6) Denote the set of corrupted voters as $\mathcal{V}_{corr}$ and the set of honest voters as $\tilde{\mathcal{V}} = \mathcal{V} \backslash \mathcal{V}_{corr}$. The game returns a bit which is 1 if and only if the following hold true:

(a) $a = a^*$

(b) $|\mathcal{V}_{corr}| \leq t$ (i.e., the number of corrupted voters is bounded by t).

Privacy: VSD **is honest:**

The e-voting system Π achieves voter privacy in case of honest VSD, for at most t corrupted voters , if there is a PPT simulator Sim such that for any PPT adversary $\mathcal{A}$:

$$|\Pr[G^{\mathcal{A},Sim}_{t-priv,\mathrm{VSD}}(1^\lambda) = 1] - \tfrac{1}{2}| = negl(\lambda)$$

5.3 Comparison with existing definitions

5.3.1 Helious. Ballot privacy.

Ballot privacy attempts to capture the idea that during its execution a secure protocol does not reveal information about the votes cast, beyond what the result of

the election leaks. In some works, ballot privacy defined even stronger: "a voter's vote is not revealed to anyone". However, in most cases ballot privacy targets specifically vote-casting procedure end

Informally, ballot privacy is satisfied if an adversary in control of arbitrarily many voters cannot distinguish between real ballots and fake ballots, where ballots are replaced by ballots for some fixed vote ϵ chosen by adversary. The adversary $\mathcal{A}$ has read access to public BB and may observe communication channels between the honest parties and BB. Note, that $\mathcal{A}$ is not allowed to corrupt any entities.

Definition: Ballot privacy for Helious:
The challenger $\mathcal{C}$ starts by flipping a coin a, which defines in what world the game between $\mathcal{C}$ and an adversary $\mathcal{A}$ would take place. If $a = 0$, the world is real, otherwise – fake. Also $\mathcal{C}$ maintains two bulletin boards $\mathrm{BB}, \mathrm{BB}'$ initialized via the setup algorithm, where BB' always contains ballots for the real votes. The adversary $\mathcal{A}$ is always given access BB and can issue two types of queries: **vote** and **ballot**. In the real world a **vote** query causes a ballot for the given vote to be placed on the both BB: hidden BB' and public BB. In the fake one, the same query causes a ballot for the given vote to be placed on the BB' and a ballot for ϵ to be placed on BB. A **ballot** query always causes the submitted ballot to be processed on both boards. At some point, the adversary $\mathcal{A}$ asks to see the result. The challenger computes tally based on BB'. The adversarial goal is to determine whether the world is real or fake.

The Ballot privacy for Helious contradicts verifiability notion by nature. Intuitively, verifiability means that it's possible to check that a vote was cast as intended, recorded as cast, tallied as recorded and, if the tally is encrypted, final result was encrypted as required. The definition states that an adversary can not distinguish real and fake world, assuming that he observes communications channels and has access to the public BB only. In general case BB and BB' contains different sets of votes, though tallying is always done using BB'. The result of an election corresponds to evaluating an arbitrary function ρ that takes a list of votes as input and returns the election result on the underlying votes. Suppose, that there is a proof π that the result was tallied as recorded. π guarantees that tallying procedure was performed on the given BB and non vote has been modified or excluded. If there is such proof, an adversary agains ballot privacy could easily check that produced result, even if decrypted correctly, was computed for

some other BB and therefore guess the challenger's coin a with an overwhelming probability.

To defend against this attack, $\mathcal{C}$ should be able to fake proof π. Suppose there is a simulator that can fake the proof π without using a global setup. That would mean that secure schemes would not satisfy tally uniqueness, since simulator allows any result to be accepted as the valid one. So , the same BB would have multiple valid election results, which contradicts verifiability.

Another case - using global setup or trapdoors.

6 Strict privacy vs E2E Verifiability

Any system Π that satisfies the definition of strict privacy, is "receipt free" but not E2E Verifiable.
The Real and Ideal executions in this case are shown on the figure 6.
Security requires existence of some simulator program $\mathcal{S}$ such that the real and ideal executions are indistinguishable for and any environment $\mathcal{Z}$. Unfortunately, as we are about to show, no such simulator exists.

Proposition:
For any system Π that satisfies the definition of strict privacy, there is an adversary $\mathcal{A}$ such that for any simulator $\mathcal{S}$, there is an environment $\mathcal{Z}$ that distinguishes ideal and real executions $\mathrm{EXEC}^{\mathcal{F},\mathcal{G}_{BB}}_{\mathcal{Z},\mathcal{S}} \not\approx \mathrm{EXEC}^{\Pi,\mathcal{G}_{BB}}_{\mathcal{Z},\mathcal{A}}$

6.1 Proof: part 1

Suppose we have a system Π, which is strictly private in case "EA is corrupted, but VSD is honest".
Consider the following attack against E2E Verifiability:

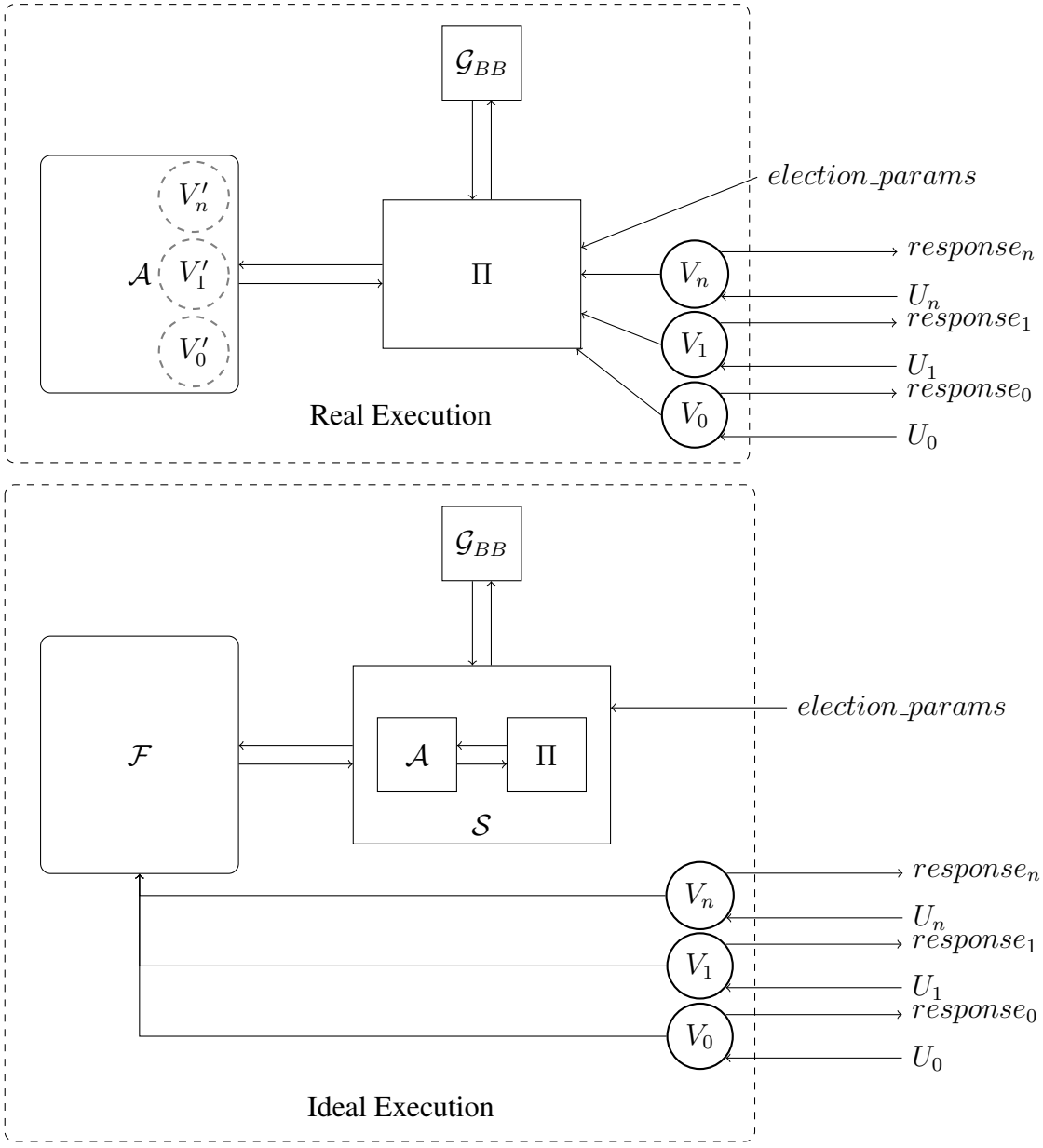


FIGURE 6: Real and Ideal Execution

$\mathcal{A}$:
– corrupts EA and VSDs but doesn't corrupt ASDs.
– corrupts t voters $(t \leq n)$, where n is the total number of voters.
– creates fake voters $\{V'_0, V'_1, \ldots, V'_n\}$ and using its power substitutes some part γ of honest voters with fake ones for the U_a option.
– fake voters vote for adversarial options according to a vote-casting procedure.
– substituted honest voters receive receipts generated for fake voters.

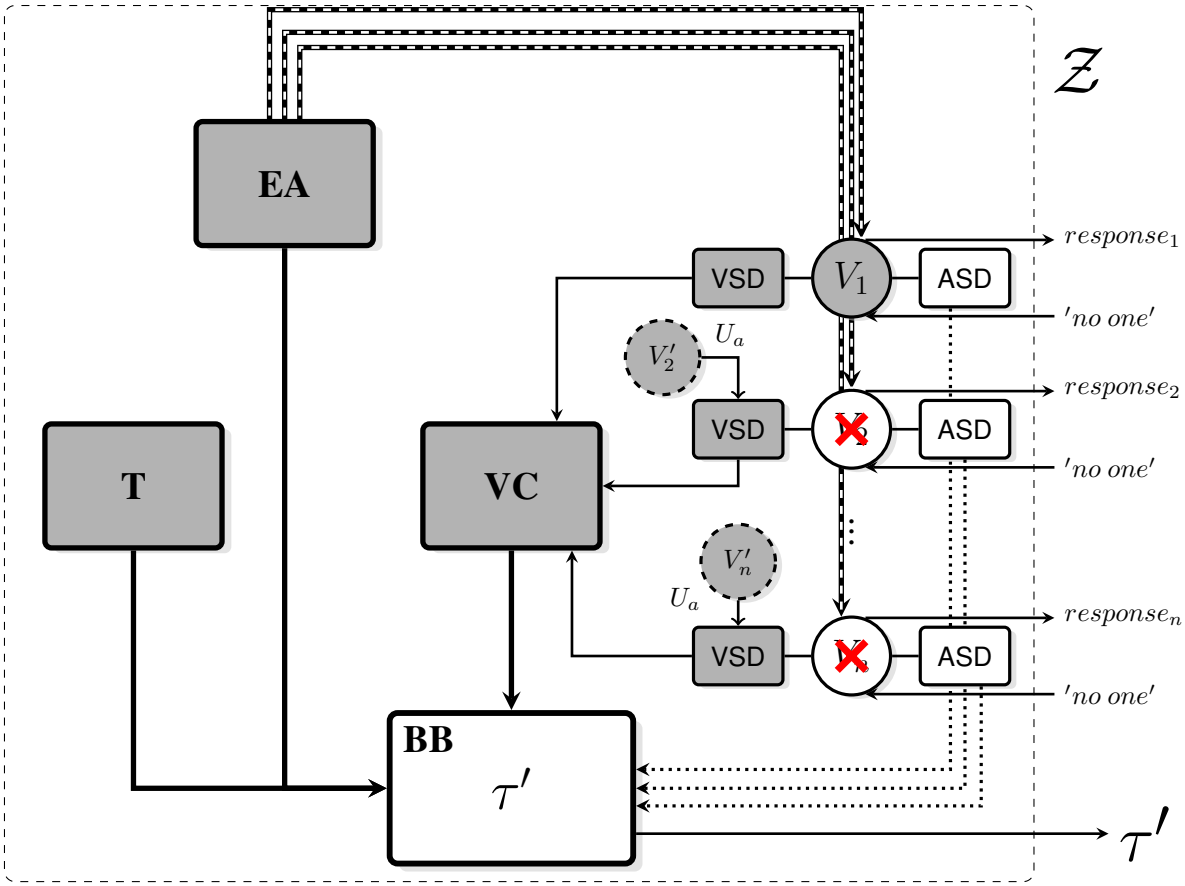


FIGURE 7: $\mathcal{A}$

let $\mathcal{Z}$ be the environment that works as follows:

$\mathcal{Z}$:
– defines an election setup information:
$election\_params = (\mathcal{C}, \mathcal{V}, \mathcal{U}, params)$, where $\mathcal{C}$ - list of candidates, $\mathcal{V}$ - list of voters, $\mathcal{U}$ - list of allowed candidates' selections, $params$ - other required information.
– instructs each voter V_i to vote for the blank '$no\ one$' option.
– stops the vote-casting phase.
– asks $\mathcal{G}_{BB}$ for the election result τ
– asks every voter V_i to verify his choice and return the result of verification - $response_i$, where $response_i$ equal to $(sid, verify\_responce, \tau')$ in case of successful verification and $\perp$ otherwise.
– If the number of successful verification responses equal to the number of voters and in all responses provided by a voter tally τ' is equal to the $\mathcal{G}_{BB}$'s tally τ return 1. Otherwise return 0.

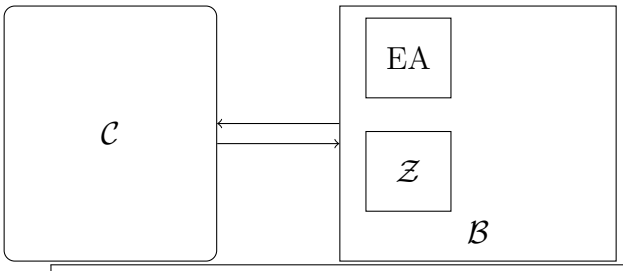
Since Π is strictly private, probability, that an adversary $\mathcal{D}$ engaging in the **Cast** protocol would distinguish real and simulated view and win the attack against the strict privacy $Pr[G^{\mathcal{D},Sim}_{strict,\mathrm{VSD},T}(1^\lambda) = 1] = \frac{1}{2} + \alpha$ where α is negligible. In case of corrupted EA, $\mathcal{C}$ returns a real ballot and receipt pair and a simulated via Sim pair. This means that in a strictly private system an adversary has a negligible chance to distinguish the ballot and receipt for his option and for some U_c option.

During the Real Execution a voter can either accept or reject the cast ballot. Rejection is possible only if he detects that the receipt and ballot are faked. The probability that a voter distinguish the receipt and ballot for his option from for the 'no-one' option and rejects is $\Pr[V_i\ rejects] \leq \alpha$, where α is negligible.

By nature of the attack $\mathcal{A}$, fake receipt is the perfectly valid receipt for the U_a option and the cast ballot would always be successfully verified. Therefore in the Real Execution $\mathcal{Z}$ would get a successful verification response from all n voters and output 1 with the probability $\Pr[\mathrm{EXEC}^{\Pi,\mathcal{G}_{BB}}_{\mathcal{Z},\mathcal{A}} = 1] = \Pr[all\ V_i\ accept] = \prod_{i=0}^{n}(1 - \Pr[V_i\ rejects]) \geq (1-\alpha)^n = 1 - n\alpha = 1 - \beta$, where $\beta = n\alpha$ is negligible. Thus, $\Pr[\mathrm{EXEC}^{\Pi,\mathcal{G}_{BB}}_{\mathcal{Z},\mathcal{A}} = 0] = 1 - \Pr[\mathrm{EXEC}^{\Pi,\mathcal{G}_{BB}}_{\mathcal{Z},\mathcal{A}} = 1] \leq \beta$, where β is negligible.

Suppose for the sake of contradiction that $\Pr[\mathrm{EXEC}^{\Pi,\mathcal{G}_{BB}}_{\mathcal{Z},\mathcal{A}} = 0] \geq \beta$, where β is non-negligible. This means that at least one voter rejects the receipt. I will show that this contradicts the definition of strict privacy. Consider an attacker $\mathcal{B}$

against strict privacy which exploits the environment $\mathcal{Z}$.



$\mathcal{B}$ interacts with the challenger in the strict privacy attack $\mathcal{C}$ as follows:
– $\mathcal{Z}$ defines an election parameters $election\_params$, assign every voter to vote for the blank option $\{V_i, `no\ one'\}$ and send this information to the $\mathcal{B}$.
– $\mathcal{B}$ forwards received $election\_params$ to the EA.
– $\mathcal{B}$ corrupts the EA.
– EA starts the election.
– $\mathcal{B}$ interacts with the challenger $\mathcal{C}$ provides $(V_i, `no\ one', U_{\mathcal{Z}_i})$ as the input.
– $\mathcal{C}$ sends back to the $\mathcal{B}$ real and simulated view $(b_0, r_0), (b_1, r_1)$ in the order defined by the challenger's coin a.
– $\mathcal{B}$ votes on behalf of all voters
– $\mathcal{Z}$ stops the vote-casting phase.
– B computes the election's tally and proof of the tally's correctness.
– $\mathcal{Z}$ asks $\mathcal{G}_{BB}$ for the election result τ
– $\mathcal{Z}$ requests every voter to verify his ballot correctness.
– $\mathcal{B}$ will run the verification on behalf of all voters using ballots and receipts from $\{b_0, r_0\}$.
– $\mathcal{Z}$ outputs 1 or 0 depending on the voters' verdict.
–$\mathcal{B}$ outputs whatever the $\mathcal{Z}$ outputs.

The challenger $\mathcal{C}$ outputs simulated ballot and receipt as (b_0, r_0), when the coin $a = 0$ and as (b_1, r_1) otherwise.

In case when $a = 0, \mathcal{B}$'s behaviour is identical to the $\mathcal{A}$'s strategy and $\mathcal{B}$ wins if $\mathcal{Z}$ outputs 0, which happens if at least one voter rejects the simulated receipt. By assumption $\Pr[\text{EXEC}^{\Pi,\mathcal{G}_{BB}}_{\mathcal{Z},\mathcal{A}} = 0] \geq \beta$, where β is non-negligible. Therefore if (b_0, r_0) is the set of simulated ballot and receipt, $\mathcal{B}$ wins with the probability $Pr[\mathcal{B} \to 0 | a = 0] = \Pr[\text{EXEC}^{\Pi,\mathcal{G}_{BB}}_{\mathcal{Z},\mathcal{A}} = 0] \geq \beta$, where β is non-negligible.

On the other hand, when $a = 1$, $\mathcal{B}$ plays honestly and the probability of $\mathcal{Z}$ out-

putting 1 is equal to probability that all voters successfully verify their votes in the honest execution, which is happens with overwhelming probability $1 - negl(\lambda)$.

The probability of $\mathcal{B}$ winning the attack against the strict privacy is $\Pr[G^{\mathcal{B}}_{strict}(1^\lambda) = 1] = \Pr[\mathcal{B} \to 0 | a = 0] \Pr[a = 0] + \Pr[\mathcal{B} \to 1 | a = 1] \Pr[a = 1] = \Pr[\mathrm{EXEC}^{\Pi, \mathcal{G}_{BB}}_{\mathcal{Z}, \mathcal{A}} = 0] \Pr[a = 0] + \Pr[\mathrm{EXEC}^{\Pi, \mathcal{G}_{BB}}_{\mathcal{Z}, honest} = 1] \Pr[a = 1] \geq \frac{1}{2}\beta + \frac{1}{2} - negl(\lambda)$, where β is not negligible. This implies that $\mathcal{B}$ wins the attack against strict privacy with the probability more than $\frac{1}{2} + negl(\lambda)$, which contradicts the assumption that the Π is strictly private.

In the Ideal Execution any simulator S can either:
1) post in $\mathcal{G}_{BB}$ the tally τ' generated by $\mathcal{A}$ or any other tally $\tau' \neq \tau$
or
2) ignore the $\mathcal{A}$'s tally and post the actual tally τ.

In the first case, the ideal functionality for E2E verifiability $\mathcal{F}$ would always detect the tally deviation caused by $\mathcal{A}$ if such exists. And since $\mathcal{A}$ doesn't corrupt ASDs, for all honest voters the ideal functionality $\mathcal{F}$ would block verification responses. This implies that in the Ideal Execution $\mathrm{EXEC}^{\mathcal{F}, \mathcal{G}_{BB}}_{\mathcal{Z}, \mathcal{S}}$ $\mathcal{Z}$ would get no response from honest voters. The total number of successful verifications would be equal to the number of corrupted voters, which is less (if not voters are corrupted) than the total number of voters – $\mathcal{Z}$ outputs 0.

In the second case, there exists a class of simulators which ignore $\mathcal{A}'$ actions and post the actual tally. For those simulators consider an modified environment $\tilde{\mathcal{Z}}$ that works as follows:

$\tilde{\mathcal{Z}}$:
Outputs a bit according to the following rules:
$\begin{cases} \text{if } \mathcal{Z} \text{ outputs 1 and the number of non-blank votes greater or equal } \gamma \text{– output 1} \\ \text{else output 0} \end{cases}$.

$\tilde{\mathcal{Z}}$ would still output 1 in case of the real execution since the number of non-blank votes would be at least γ due to successful attack $\mathcal{A}$. However $\tilde{\mathcal{Z}}$ would not find at least γ non-blank votes and output 0 in the ideal execution.

Thus, there is the attacker $\mathcal{A}$ such that for any simulator $\mathcal{S}$ there is the environment $\tilde{\mathcal{Z}}$ or $\mathcal{Z}$ which can always distinguish real and ideal executions.

6.2 Proof: part 2

Suppose we have a system Π', which is strictly private in case "EA is honest, but VSD is corrupted".
Consider the following attack against E2E Verifiability:

$\mathcal{A}'$:
– corrupts EA and VSDs but doesn't corrupt ASDs.
– chooses an option $U_a \in \mathcal{U}$
– corrupts t voters $(t \leq n)$, where n is the total number of voters.
– provides every honest voter V_i with fake credentials s'_i s.t. the ballot and receipt produced using $(s'_i, `no\ one')$ are identical to the ballot and receipt produced using real credentials for an option $U_a$: $b, r \leftarrow (s'_i, `no\ one')$ AND $b, r \leftarrow (s_i, U_a)$
– creates fake voters $\{V'_0, V'_1, \ldots, V'_n\}$ and provides them with real credentials s_i
– using its power substitutes some part γ of honest voters' cast protocols with the fake voters' protocols for the U_a option.

let $\mathcal{Z}'$ be the environment that works as follows:

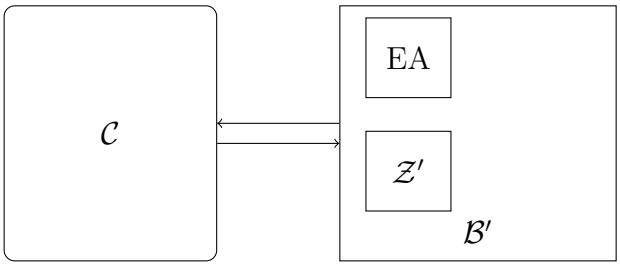
$\mathcal{Z}'$:
– defines an election setup information:
$election\_params = (\mathcal{C}, \mathcal{V}, \mathcal{U}, params)$, where $\mathcal{C}$ - list of candidates, $\mathcal{V}$ - list of voters, $\mathcal{U}$ - list of allowed candidates' selections, $params$ - other required information.
– instructs each voter V_i to vote for the blank option ('no one').
– stops the vote-casting phase.
– asks $\mathcal{G}_{BB}$ for the election result τ
– asks every voter V_i to verify his choice and return the result of verification - $response_i$, where $response_i$ equal to $(sid, verify\_responce, \tau')$ in case of successful verification and $\perp$ otherwise.
– If the number of successful verification responses equal to the number of voters and in all responses provided by a voter tally τ' is equal to the $\mathcal{G}_{BB}$'s tally τ return 1. Otherwise return 0.

Since Π' is strictly private, probability, that an adversary engaging in the **Reg-**

istration protocol would distinguish real and simulated view and win the attack against the strict privacy $|\Pr[G^{\mathcal{A},Sim}_{strict,\mathrm{EA}}(1^\lambda)=1]-\frac{1}{2}|=\alpha$ where α is negligible. In case "EA is honest, but VSD is corrupted", simulated view is credentials $\tilde{s_i}$ generated via the simulator Sim.

In the Real Execution a voter V_i starts the Cast protocol using fake credentials s'_i and an option '$no\ one'$. However, $\mathcal{A}'$ substitutes his ballot and receipt with the ballot and receipt generated for a fake voter with real credentials and U_a option. By nature of the attack, the returned receipt generated for real credentials and the U_a option is identical to the receipt produced for a fake credentials and '$no\ one'$ option. Therefore in the Real Execution $\mathcal{Z}'$ would output 1 if non of the voters would detect that he was given a receipt for fake credentials. The probability of this event is $\Pr[\mathrm{EXEC}^{\Pi',\mathcal{G}_{BB}}_{\mathcal{Z}',\mathcal{A}'}=1]=\Pr[all\ V_i\ accept]=\prod_{i=0}^{n}(1-\Pr[V_i\ rejects])$. Suppose, the probability of rejection by V_i is $\Pr[V_i\ rejects]=\zeta$. Thus, $\Pr[\mathrm{EXEC}^{\Pi',\mathcal{G}_{BB}}_{\mathcal{Z}',\mathcal{A}'}=1]=(1-\zeta)^n=1-n\zeta=1-\zeta'$.

Suppose that ζ' is not negligible. This means that $\Pr[\mathrm{EXEC}^{\Pi',\mathcal{G}_{BB}}_{\mathcal{Z}',\mathcal{A}'}=0]=1-\Pr[\mathrm{EXEC}^{\Pi',\mathcal{G}_{BB}}_{\mathcal{Z}',\mathcal{A}'}=1]=\zeta'$, where ζ' is not negligible. This means that at least one voter rejects the receipt with a non-negligible probability. We will show that this contradicts the definition of strict privacy. Consider an attacker $\mathcal{B}'$ against strict privacy which exploits the environment $\mathcal{Z}'$.



$\mathcal{B}'$ interacts with the challenger in the strict privacy attack $\mathcal{C}$ as follows:
– $\mathcal{Z}$ defines an election parameters $election\_params$, assign each voter the blank option $\{V_i, `no\ one'\}$ and send this information to the $\mathcal{B}$.
– $\mathcal{B}'$ forwards received $election\_params$ to the $\mathcal{C}$.
– $\mathcal{B}'$ corrupts the VSD.
– $\mathcal{C}$ starts the election.
– $\mathcal{B}'$ sends to $\mathcal{C}$ $(V_i, `no\ one', U_a)$.
– $\mathcal{C}$ sends back s_0, s_1.
– $\mathcal{B}'$ provides voters with credentials s_0 and uses $(s_0, `no\ one')$ to produce the ballot and receipt and posts the result on BB.
– $\mathcal{Z}'$ stops the vote-casting phase.
– $\mathcal{C}$ executes the $Tally$ protocol.
– $\mathcal{Z}'$ asks $\mathcal{G}_{BB}$ for the election result τ
– $\mathcal{Z}'$ requests every voter to verify his ballot correctness.
– $\mathcal{B}'$ will run the verification on behalf of all voters.
– $\mathcal{Z}'$ outputs 1 or 0 depending on the voters' verdict.
– $\mathcal{B}'$ outputs whatever the $\mathcal{Z}'$ outputs.

In case when the simulated credentials are chosen ($a = 0$), $\mathcal{B}'$ provides a voter with fake credentials and generated ballot and receipt for the blank option using the fake credentials, which means that real credentials correspond to the U_a option. $\mathcal{B}'$'s behaviour is identical to the $\mathcal{A}'$'s strategy. $\mathcal{B}'$ wins if outputs 0, which happens if $\mathcal{Z}'$ outputs 0. $\Pr[\mathcal{B}' \to 0 | a = 0] = \Pr[\mathrm{EXEC}^{\Pi', \mathcal{G}_{BB}}_{\mathcal{Z}', \mathcal{A}'} = 0] = \zeta'$

Else if $a = 1$, $\mathcal{B}'$ provides a voter with real credentials and uses the real credentials to vote for the blank option, which is honest behaviour. $\Pr[B' \to 1 | a = 1] = \Pr[\mathrm{EXEC}^{\Pi', \mathcal{G}_{BB}}_{\mathcal{Z}', honest} = 1] = 1$

Thus, the probability of $\mathcal{B}'$ winning the attack against the strict privacy is $\Pr[G^{\mathcal{B}, Sim}_{strict, \mathrm{EA}}(1^\lambda) = 1] = \frac{1}{2}(\Pr[B' \to 0 | a = 0] + \Pr[B' \to 1 | a = 1]) = \frac{1}{2}(\zeta' + 1) = \frac{1}{2} + \frac{1}{2}\zeta'$, where ζ' is not negligible. This implies that $\mathcal{B}'$ wins the attack against strict privacy with the probability more than $\frac{1}{2} + negl(\lambda)$, which contradicts the assumption that the Π is strictly private.

In the Ideal Execution any simulator S can either:
1) post in $\mathcal{G}_{BB}$ the tally τ' generated by $\mathcal{A}'$ or any other tally $\tau' \neq \tau$
or

2) ignore the $\mathcal{A}'$'s tally and post the actual tally τ.

In the first case, the ideal functionality for E2E verifiability $\mathcal{F}$ would always detect the tally deviation caused by $\mathcal{A}'$ if such exists. And since $\mathcal{A}'$ doesn't corrupt ASDs, for all honest voters the ideal functionality $\mathcal{F}$ would block verification responses. This implies that in the Ideal Execution $\mathrm{EXEC}^{\mathcal{F},\mathcal{G}_{BB}}_{\mathcal{Z}',\mathcal{S}}$ $\mathcal{Z}'$ would get no response from honest voters. The total number of successful verifications would be equal to the number of corrupted voters, which is less (if not voters are corrupted) than the total number of voters – $\mathcal{Z}'$ outputs 0.

In the second case, there exists a class of simulators which ignore $\mathcal{A}'$ actions and post the actual tally. For those simulators consider an modified environment $\tilde{\mathcal{Z}}'$ that works as follows:

$\tilde{\mathcal{Z}}'$:
Outputs a bit according to the following rules:
$\begin{cases} \text{if } \mathcal{Z} \text{ outputs 1 and the number of non-blank votes greater or equal } \gamma\text{– output 1} \\ \text{else output 0} \end{cases}$.

$\tilde{\mathcal{Z}}'$ would still output 1 in case of the real execution since the number of non-blank votes would be at least γ due to successful attack $\mathcal{A}$. However $\tilde{\mathcal{Z}}'$ would not find at least γ non-blank votes and output 0 in the ideal execution.

Thus, there is the attacker $\mathcal{A}'$ such that for any simulator $\mathcal{S}$ there is the environment $\tilde{\mathcal{Z}}'$ or $\mathcal{Z}'$ which can always distinguish real and ideal executions.

7 Blind signature e-voting scheme

Here is the description of a blind signature e-voting system for 1-out-of-m elections.
The Blind signature scheme **Setup** ($1^\lambda, \mathcal{P} = \{P_1, \ldots, P_m\}, \mathcal{V} = \{V_1, \ldots, V_n\}, \mathcal{U} = \{\{P_1\}, \ldots, \{P_m\}\}$)**.**

Let $(GenBL, EncrB, SignBL)$ be the PPT algorithms that constitutive the RSA blind signature scheme, PRG_{str} - a function for pseudo random string generation, PRG_{prime} - a function for pseudo-random prime number generation and $(Gen, Encr, Decr)$ - PPT algorithms for the ELGamal encryption scheme. The EA runs $GenBL(Param, 1^\lambda)$ to generate the blind signature scheme keys (bsk, bpk)

and $Gen(Param, 1^\lambda)$ to generate ELGamal keys (sk, pk). Public keys bpk, pk and functions $EncrB, Encr, PRG_{str}, PRG_{prime}$ are posted on the BB.
Then, for every voter V_l, where $l \in [n]$, EA runs $PRG_{str}(1^\lambda)$ function to generate random string $seed_l$.

Registration

Let $(GenAES, EncrAES, DecrAES)$ be the publicly known PPT algorithms that implements AES encryption scheme and $Hash$ - hash algorithm.

Every voter V_l completes the following procedure:
– uses the published on the BB function $PRG_{str}(1^\lambda)$ to generate his alias x_l and $PRG_{prime}(bpk)$ to generate a blinding factor z_l.
– calculates value $p_l = Hash(x_l)EncrB(z_l)$.
– chooses his secret password $password_l$ and runs $GenAES(password_l, s_l)$ to generate his key for symmetric encryption key_l, where $s_l = PRG_{str}(seed_l)$.
– encrypts x_l and z_l by running $EncrAES(key_l, x_l)$ and $EncrAES(key_l, z_l)$ respectively to get encrypted values $\hat{x}_l, \hat{z}_l$
– sends $p_l, \hat{x}_l, \hat{z}_l$ to the EA

EA:
Upon receiving $p_l, \hat{x}_l, \hat{z}_l$ from a voter, EA posts all information to the BB.
When registration is closed, EA runs $SignBL(bsk, p_l)$ for every entry in the BB and post the result in the corresponding line as p_l^{sign}.

Cast:
Let $e_l = (e_{1_l}, e_{2_l}, \ldots, e_{m_l})$ be the characteristic vector corresponding to the voter's selection, where $e_{j_l} = 1$ if the option opt_j is selected by the voter V_l.
Every voter V_l completes the following procedure:
– gets all information from the BB - $\{p_l, \hat{x}_l, \hat{z}_l\}$.
– finds his entry and decrypts x_l and z_l by running $DecrAES(key_l, \hat{x}_l)$ and $DecrAES(key_l, \hat{z}_l)$ respectively, where $key_l = GenAES(password_l, s_l)$ and $s_l = PRG_{str}(seed_l)$
– computes σ_l - signature for x_l by calculating $p_l^{signed} z_l^{-1}$
– computes Π_l – NIZK proof of signature knowledge .
– chooses his vote-option e_{j_l} and writes the corresponding characteristic vector e_l
– for $j \in [m]$ compute $c_{j_l} = Encr(pk, e_{j_l})$
– computes NIZK proofs π_{j_l} that each c_{j_l} is an encryption of 1 or 0.
– sends $b_l = (x_l, \sigma_l, \Pi_l, c_l, \pi_l)$ to the EA.

– keeps x_l as receipt.
– If EA accept b_l, protocol terminates successfully.
*Optional: every voter can export the randomness to check that the ballot was cast as intended.

Upon receiving $(x_l, \sigma_l, \Pi_l, c_l, \pi_l)$ from a voter, EA checks NIZK proofs and, if it's valid, accepts the ballot and posts all information to the BB.

Tally:
After election is closed, EA computes $\mathcal{C}$ – the sum of all c_l and runs $Decr(sk, \mathcal{C})$ to decrypt Tally τ. EA posts the τ along with the proof of tally correctness $Proof$.

Result:
result is straightforward.

Verify:
The algorithm returns 1 only if the following checks are true:
– exported randomness is correct or voter choose not to check.
– there is ballot with x_l
– all Π_l, π_l are valid
– number of ballots less or equal to the number of registered voters
– $Proof$ is correct
– sum of all scores at τ are less than or equal to ballot numbers

Construction of the vote extractor.
$\mathcal{E}$ has input τ and the set of receipts $\{x_l\}_{V_l \in \tilde{\mathcal{V}}}$ where $\tilde{\mathcal{V}}$ is the set of the honest voters that voted successfully. If $Result(\tau) = \perp$ (i.e., the transcript is not meaningful), then $\mathcal{E}$ outputs $\perp$. Otherwise, $\mathcal{E}$ (arbitrarily) arranges the voters in $\mathcal{V} \backslash \tilde{\mathcal{V}}$ and the tags not included in $\{x_l\}_{V_l \in \tilde{\mathcal{V}}}$ as $\langle V_l^{\mathcal{E}} \rangle_{l \in [n - |\tilde{\mathcal{V}}|]}$ and $\langle tag_l^{\mathcal{E}} \rangle_{l \in [n - |\tilde{\mathcal{V}}|]}$ respectively. Next, for every $l \in [n - |\tilde{\mathcal{V}}|]$:

(1) $\mathcal{E}$ finds at the BB entry with $x_l = tag_l^{\mathcal{E}}$ and brute-force the corresponding ELGamal cipher to open the selected candidate $\mathcal{P}_l^{\mathcal{E}}$. If $\mathcal{P}_l^{\mathcal{E}}$ is the valid candidate's selection, then $\mathcal{E}$ sets $\mathcal{U}_l^{\mathcal{E}} = \{\mathcal{P}_l^{\mathcal{E}}\}$. Otherwise it inputs $\perp$.

Finally $\mathcal{E}$ outputs $\langle \mathcal{U}_l^{\mathcal{E}} \rangle_{V_l \in \mathcal{V} \backslash \tilde{\mathcal{V}}}$