Adaptively Secure Single Secret Leader Election from DDH

Dario Catalano[1], Dario Fiore[2], Emanuele Giunta[2,3]

[1] University of Catania, Italy.
catalano@dmi.unict.it
[2] IMDEA Software Institute, Madrid, Spain.
{dario.fiore, emanuele.giunta}@imdea.org
[3] Universidad Politecnica de Madrid, Spain.

Abstract. Single Secret Leader Election protocols (SSLE, for short) allow a group of users to select a random leader so that the latter remains secret until she decides to reveal herself. Thanks to this feature, SSLE can be used to build an election mechanism for proof-of-stake based blockchains. In particular, a recent work by Azouvi and Cappelletti (ACM AFT 2021) shows that in comparison to probabilistic leader election methods, SSLE-based proof-of-stake blockchains have significant security gains, both with respect to grinding attacks and with respect to the private attack. Yet, as of today, very few concrete constructions of SSLE are known. In particular, all existing protocols are only secure in a model where the adversary is supposed to corrupt participants before the protocol starts – an assumption that clashes with the highly dynamic nature of decentralized blockchain protocols.
In this paper we make progress in the study of SSLE by proposing new efficient constructions that achieve stronger security guarantees than previous work. In particular, we propose the first SSLE protocol that achieves adaptive security. Our scheme is proven secure in the universal composability model and achieves efficiency comparable to previous, less secure, realizations in the state of the art.

A short version of this work appears in the proceedings of PODC '22, https://doi.org/10.1145/3519270.3538424

Table of Contents

1 Introduction

Electing a leader is a matter of central importance to realize distributed consensus. Over recent years, the growing diffusion of blockchain related technologies sparkled renewed interest on this problem, both from an industrial and an academic perspective. Also, it motivated the need to add privacy properties to consensus protocols and applications. For instance, in the case of Proof of Stake blockchains (e.g. [BGM16, BPS16]) it is often desirable to have an election procedure where the identity of the randomly selected leader remains secret until the latter reveals herself [AMM18, GHM+17, GOT19, KKKZ19]. This secrecy feature provides an elegant way to strengthen the liveness of the blockchain against several attacks. For instance, it allows to prevent Denial of Service attacks against the chosen leader, that could otherwise prevent the latter to publish her block. Most common realizations of secret leader elections, however, manage to elect a leader in expectation (e.g. [BGM16, BPS16]). In a nutshell, probabilistic leader elections are protocols where, at any given round, a single leader is likely elected but it could be the case that more (or zero) leaders are elected. This causes potential wasted efforts and might lead to forks in the blockchain.

This motivates the study of Single Secret Leader Election (SSLE) procedures that guarantee that one single leader per round is actually elected [Lab19]. This feature makes SSLE a better candidate for building proof-of-stake based consensus mechanisms. Indeed, a recent result of Azouvi and Cappelletti [AC21] shows that (secretly) electing exactly one leader per round leads to significant security gains (compared to probabilistic leader election schemes), both with respect to grinding attacks and with respect to the private attack.

When it comes to constructing SSLE protocols, a (theoretically) simple way to realize it is via secure multiparty computation. However, such solutions would typically require all parties to send/receive messages, which is not scalable in a blockchain scenario where there are many parties and the adversary can take one or more of them offline. Ideally, beyond being computationally efficient, an SSLE should require low communication complexity and very little interaction: once a player registers, she should be able to participate to several election rounds without requiring any significant further interaction.

The question of realizing SSLE was recently addressed in [BEHG20, CFG21]. In [BEHG20] Boneh, Eskandarian, Hanzlik and Greco formalized the notion of SSLE as a distributed protocol that (secretly) elects a leader according to the following rules: one single leader per round is elected (uniqueness), all parties have the same chances to become leaders (fairness) and nobody should be able to guess the next leader better than at random (unpredictability). Boneh, Eskandarian, Hanzlik and Greco also proposed three realizations of SSLE achieving very different efficiency guarantees and tradeoffs. The first two solutions are asymptotically efficient, but rely on rather expensive building blocks (indistinguishability obfuscation [BGI+01, GGH+13] and threshold fully-homomorphic encryption [BGG+18], respectively). The third solution is asymptotically less efficient but it relies on practical building blocks. In a nutshell, their protocol consists in shuffling n Diffie-Hellman pairs (one for each participant). The construction from [CFG21] proposes a different approach to the problem: it puts forward a stronger universally composable security definition of SSLE and a concrete construction realizing this notion. The approach to the construction builds on Public Key Encryption with Keyword Search (PEKS) [BDOP04, ABC+05], and they show an efficient protocol based on pairings, which is the first to achieve so-called *on chain efficiency*: the number of bits to store on chain, at each round, is at most $O(\log^2 n)$ (a property that was achieved only by the iO-based scheme of [BEHG20]).

A common limitation of all the existing SSLE constructions, however, is that they can be proved secure only with respect to *static* adversaries, i.e., adversaries that are forced to choose who to corrupt at the beginning of the protocol. This is a severe limitation in the highly dynamic contexts of blockchain applications, and it leaves unanswered the question of realizing a practical SSLE protocol secure against more realistic adaptive adversaries.

Our Contribution.

In this paper we propose the first SSLE protocol that achieves (universally composable) security in the presence of adaptive corruptions. To obtain this result, we revisit the DDH-based SSLE realization from [BEHG20]. First, we propose a variant of this scheme, see Section 3.2, that achieves a stronger security guarantee – universal composability against static adversaries – and roughly halves the communication cost. Second, we give our main result, which is a novel SSLE protocol that achieves adaptive security with erasures, while remaining both universally composable (according to the definition from [CFG21]) and efficient. In what follows we give an informal description of our protocols.

As a warm up, we present the simpler construction achieving static security only. Recall that Boneh, Eskandarian, Hanzlik and Greco's protocol consists of maintaining an initially empty list l of commitments on the blockchain. A new user registers by choosing a random secret $x_i \in \mathbb{Z}_q$, adding $\mathsf{Com}(x_i, r_i) = (g^{r_i}, g^{x_i r_i})$ for some random r_i to l, and shuffling (i.e., randomly permuting and re-randomizing) the updated list. To avoid subtle attacks, see Appendix A.2 for details, the shuffler should also provide a non-interactive zero-knowledge (NIZK) proof π that the shuffle was performed correctly. The total (on chain) communication costs per player are thus $2n$ group elements + $|\pi|$. Elections are carried out by choosing a commitment in l through a random beacon, a primitive that returns unbiased randomness, and the winner can claim victory by revealing the secret x_i of the chosen commitment.

Our base construction builds on the observation that instead of adding a new pair $(g^{r_i}, g^{x_i r_i})$ to the list, parties can "share between them" the random component g^r of the commitment and simply provide $g^{r x_i}$ (proving knowledge of the secret x_i). This change alone allows us to reduce the communication complexity of the scheme to $n + 1$ group elements (plus the cost of the NIZK). Moreover, once a party's commitment is chosen during an election, instead of revealing the secret x_i, she proves knowledge of x_i and that the same x_i was used to generate her initial commitment. Since no secret value is ever revealed, this grants us the benefit of making UC security easy to prove and, indeed, the only explicitly UC-secure component needed by our protocol is a proof of knowledge at registration time. Moreover, up to minor adaptations, this protocol also achieves adaptive security albeit only with respect to a game-based definition a-la [BEHG20, CFG21] (i.e., appropriately modified to consider adaptive adversaries). Informally, this comes from the fact that commitments of yet uncorrupted players are actually indistinguishable from random elements from the adversary's perspective and corrupting a new party does not reveal any information about not yet corrupted ones.

This feature is not enough to achieve simulation based security, though. Indeed, at election time the simulator would have to indicate some random commitment g^{rx} linked to some honest user whenever the functionality $\mathcal{F}_{\mathsf{SSLE}}$ says that an uncorrupted user won[4]. This is problematic as, if

[4] We recall here that according to the definition from [CFG21], the simulator learns the actual winner only after $\mathcal{F}_{\mathsf{SSLE}}$ sends some results message

subsequently some party P is corrupted, the simulator needs to provide a secret exponent x that is consistent with all previous elections won by P (and such an exponent might well not even exist!).

In principle this issue, related to the well known *selective decommitment problem* [DNRS03], could be addressed via sophisticated tools such as non-committing encryption, which however (in the standard model) would render our solution impractical (beyond requiring more interaction). Instead, we develop a simpler solution, that consists in replacing the commitments discussed above with ones that allow for randomizable openings, so to make the keys updatable. Periodically, users securely refresh their commitments/keys so to make them unlinkable to their own previous commitments/keys. A bit more concretely, we let user's commitments be of the form (g_1^r, g_2^r, h) where the first couple is shared among all commitments in l and $h = g_1^{rx} g_2^{r\delta}$ (the index δ being publicly linked to P at registration time).

A first attempt to perform key update consists in letting each party P choose a random ω, post a key update $u = g_1^{r\omega}$, so that her commitment becomes $h \cdot u = g_1^{r(x+\omega)} g_2^{r\delta}$, store the new key $(x + \omega, \delta)$, and *erase* the old one. Notice that the erasing step is important here to enable the unlinkability desiderata mentioned above against adversaries that may adaptively corrupt this user at a later point in time.

Unfortunately, however, this simple solution does not work because of shuffling. Recall that our protocol dictates that the h's are randomly shuffled at each new election, meaning that once the element u is provided, parties won't know to which commitment in the list l they should apply it to.

We fix this by maintaining *two* different lists of commitments, consisting of n entries each. The first list contains the elements h discussed above and is shuffled frequently whereas the second list $l' = \{k_\delta\}_{\delta \in [n]}$ is never shuffled. In each round, party P, owning key (x, δ) is linked to a commitment h if $g_1^{rx} g_2^{r\delta} = hk_\delta$. The key difference is that now updates u can be applied to the component k_δ of the commitment that is publicly associated to P even though the h component remains unlinked. This simple trick allows parties to securely update their key and, by erasing outdated ones, to prove security against adaptive adversaries.

In conclusion, the communication costs for each shuffle are $2n + 2$ group elements (+ the size $|\pi|$ of the proof of correct shuffle) plus the costs associated to each user to update her key (which in our case is a single group element per user). In Section 4.3 we provide some optimizations to reduce the amortized communication to $2n + \Theta(\sqrt{n \log n})$ group elements per shuffle, matching the DDH-based Boneh et al. solution which achieves only static, game-based security.

Related work

The problem of single secret leader election (SSLE) was first considered in an RFP by Protocol Labs [Lab19], which also informally describes a solution based on functional encryption [BSW11]. Boneh et al. [BEHG20] formalized the notion of SSLE proposed three constructions that we briefly discussed earlier. In a more recent work, Catalano, Fiore and Giunta [CFG21] proposed a UC-secure definition of SSLE and showed an efficient construction that achieves on-chain efficiency.

If we drop the requirement of electing a *single* leader, other works have considered the problem of keeping the leader secret in proof-of-stake based protocols, e.g., [BGM16, BPS16, GOT19, KKKZ19]. Another approach to probabilistic secret leader election is that of Algorand [GHM$^+$17] and Fantomette [AMM18] based on verifiable random functions (VRFs). The VRF is used to identify a few potential leaders that must manifest and then proceed to choose a winner among them by using some tie-breaker method (e.g., smallest VRF output). The drawback of this approach is that the final leader does not know she is the leader until all the potential leaders publish their values. In addition, the users that do not receive the winner's output might incorrectly believe that a different leader was elected, which can lead to a fork in the chain. Such uncertain situations cannot occur when a single leader is elected, as in SSLE. This is why SSLE can lead to more secure solutions as recently confirmed by Azouvi and Cappelletti [AC21] who show that single leader election achieves higher security gains than probabilistic election methods, both when considering the private attack (the worst attack on longest-chain protocols [DKT$^+$20]) and grinding attacks.

2 Preliminaries

2.1 Notation

$\lambda \in \mathbb{N}$ denotes the security parameter and a function $\varepsilon : \mathbb{N} \to \mathbb{N}$ is called negligible if it vanishes faster than λ^{-c} for any constant c. $[n] = \{0, \dots, n-1\}$ is the set of natural number smaller than n. S_n is the substitution group, i.e. the set of all bijections $\eta : [n] \to [n]$.

$\mathbb{G}$ is a group of prime order $q = 2^{O(\lambda)}$ whose operation are expressed multiplicatively, g a canonical generator, and $\mathbb{F}_q$ the finite field of order q. Bold font $(\mathbf{g}, \mathbf{h}, \mathbf{z}, \dots,)$ is reserved for vectors of either group of field elements. Given $\mathbf{g} \in \mathbb{G}^n$ and $\mathbf{x} \in \mathbb{F}_q^n$ we denote the multi-exponentiation as $\mathbf{g}^{\mathbf{x}} = g_1^{x_1} \cdot \dots \cdot g_n^{x_n}$. We assume that the DDH problem is hard in $\mathbb{G}$, i.e. for any PPT algorithm $\mathcal{A}$

$$\left| \Pr\left[\mathcal{A}(g, g^x, g^y, g^{xy}) \to 1\right] - \Pr\left[\mathcal{A}(g, g^x, g^y, g^r) \to 1\right] \right| \leq \varepsilon(\lambda)$$

with ε negligible, and x, y, r are uniformly sampled over $\mathbb{F}_q$. In our protocols, N denotes the total number of users and n the number of *registered* users (over the various executions). For a given finite set X, $x \leftarrow^{\$} X$ means that x is sampled uniformly from X with fresh random coins. Similarly, when a random variable y is uniformly distributed over X we write $y \sim U(X)$.

2.2 Non Interactive Zero-Knowledge Arguments

A non-interactive zero-knowledge (NIZK) argument for a given relation $\mathcal{R}$ is a tuple of PPT algorithms $(\mathsf{NIZK.G}, \mathsf{NIZK.P}, \mathsf{NIZK.V})$ such that: $\mathsf{NIZK.G}$ initialise the common reference string crs; for any $(x, w) \in \mathcal{R}$ the prover produces a proof $\pi \leftarrow^{\$} \mathsf{NIZK.P}(\mathsf{crs}, x, w)$; for any statement x and proof π, the verifier $\mathsf{NIZK.V}(\mathsf{crs}, x, \pi)$ outputs either 1, meaning that the proof is accepted, or 0 in case it is rejected.

A NIZK is correct if for $(x, w) \in \mathcal{R}$, $\mathsf{crs} \leftarrow^{\$} \mathsf{NIZK.G}(1^\lambda)$ and $\pi \leftarrow^{\$} \mathsf{NIZK.P}(\mathsf{crs}, x, w)$ the verifier accepts, i.e. $\mathsf{NIZK.V}(\mathsf{crs}, x, \pi)$ returns 1. In our construction we also require NIZKs to be weakly simulation extractable [Sah99] and zero-knowledge [FLS90]. Informally, weak simulation extractability assumes the existence of an extractor able to derive a witness w from proofs created by an adversary that are different from previously generated, possibly simulated, ones. A standard approach then to make proofs unique is to add prover's and session's ID to the statement – which for instance in the Fiat-Shamir transform translates into salting the hash function with this information.

In our protocol we extensively use NIZKs to prove statements related to knowledge of the discrete logarithm for given group elements, and the correctness of shuffling. Concretely, we will considers NIZKs for the relations liste below. At a very high level, $\mathcal{R}_{\mathsf{DH}}$ asks that a given tuple of element (g, h, u, v) is Diffe-Hellman, i.e. that there exists an x such that $h = g^x$ and $v = u^x$. $\mathcal{R}_{\mathsf{ped}}$ associates a Pedersen commitment [Ped92] h and a given value δ to the randomness α that opens h to δ. Next,

$\mathcal{R}_{\mathsf{sh}}$ asks if given two vectors in $\mathbb{G}^n$, the second one is obtained by randomizing and shuffling the first. This relation is generalized by $\mathcal{R}_{\mathsf{Esh}}$ that further checks that certain given elements $\mathbf{k}$ were exponentiated with the same randomness of the shuffled ones.

$$\begin{aligned}
\mathcal{R}_{\mathsf{DH}} &= \{((g,h,u,v),x) \ : \ h = g^x, \ v = u^x\} \\
\mathcal{R}_{\mathsf{ped}} &= \{((\mathbf{g},h,\delta),\alpha) \ : \ \mathbf{g} \in \mathbb{G}^2, \ h \in \mathbb{G}, \ h = \mathbf{g}^{(\alpha,\alpha\delta)}\} \\
\mathcal{R}_{\mathsf{sh}} &= \{((g,\mathbf{h},\widetilde{g},\widetilde{\mathbf{h}}),(r,\eta)) \ : \ r \in \mathbb{F}_q, \ \eta \in S_n, \ \widetilde{g} = g^r, \ \widetilde{h}_j = h^r_{\eta(j)}\} \\
\mathcal{R}_{\mathsf{Esh}} &= \{((\mathbf{g},\mathbf{h},\mathbf{k},\widetilde{\mathbf{g}},\widetilde{\mathbf{h}},\widetilde{\mathbf{k}}),(r,\eta)) \ : \ r \in \mathbb{F}_q, \ \eta \in S_n, \ \widetilde{\mathbf{g}} = \mathbf{g}^r, \ \widetilde{h}_i = h^r_{\eta(i)}, \ \widetilde{k}_i = k^r_i\}.
\end{aligned}$$

2.3 UC framework

The celebrated UC framework [Can01] allows to define and prove strong security properties of a protocol which will be preserved under parallel or nested composition. Central in this model is the notion of *ideal functionality* $\mathcal{F}$ which captures the intended behaviour of a given protocol Π. Π is said to UC-realise $\mathcal{F}$ if there exists a simulator $\mathcal{S}$ such that Π is indistinguishable from $\mathcal{F} \circ \mathcal{S}$. The role of $\mathcal{S}$ is to emulate messages that would be sent by honest parties in Π while interacting with $\mathcal{F}$ on behalf of malicious ones.

Notably the adversary or *environment* $\mathcal{Z}$ in the indistinguishability experiment manages honest users' inputs and privately receives their output while $\mathcal{S}$ has no access to them. Moreover, in the active setting, $\mathcal{Z}$ can adaptively corrupt any number of parties, learning their internal state which may have to be simulated by $\mathcal{S}$, and influence their behaviour.

A protocol Π can run an ideal functionality $\mathcal{H}$ as a subroutine, in which case it is called $\mathcal{H}$-hybrid. Security of Π can then argued in the $\mathcal{H}$-hybrid model instead of replacing $\mathcal{H}$ with a protocol that UC-realises it. Our construction will implicitly be in the $\mathcal{F}_{\mathsf{AuBr}}$-hybrid model, where $\mathcal{F}_{\mathsf{AuBr}}$ is an authenticated broadcast communication channel.

Functionalities. In Fig. 1 we present two functionalities extensively used in our protocols: $\mathcal{F}^{\mathcal{R}}_{\mathsf{zk}}$ for UC zero-knowledge and $\mathcal{F}_{\mathsf{ct}}$ for a random beacon for publicly-verifiable coin tossing. The first is defined and realized in [CF01], even though our definition make all users receive the output messages. $\mathcal{F}_{\mathsf{ct}}$ instead was first introduced and realized in [CD20] assuming an honest majority under standard assumptions. In practical application it may be realised by taking the hash of an unpredictable value parties have agreed on. We remark that our use of the random oracle is justified assuming a global RO functionality in the GUC model [CJS14, CDG$^+$18]. Finally, about our communication model, we assume an authenticated broadcast channel with known bounded delay [KMTZ13], which implies that messages sent in broadcast are eventually delivered (not necessarily in the right order). Although this introduce some degree of synchronicity, this is in line with previous work [BEHG20, CFG21].

Secure Erasures. A significant downside of the UC framework is that security if often hard or impossible to achieve, due to the limited power granted to $\mathcal{S}$, and even more so when $\mathcal{Z}$ performs adaptive corruptions. For instance, due to the *selective decommitment problem* introduced in [DNRS03] several impossibility results are known about realising non-interactive UC-encryption schemes secure against adaptive corruptions in the standard model, [Hof11]. Even though non-committing encryption [CFGN96] bypasses this impossibility adding more rounds, all currently known schemes induce significant efficiency overheads.

The Zero-Knowledge Functionality $\mathcal{F}_{\mathsf{zk}}^{\mathcal{R}}$:
Upon receiving $(\mathsf{prove}, sid, x, w)$ from P_i, with sid being used by P_i for the first time: if $(x, w) \in \mathcal{R}$, broadcast $(\mathsf{proof}, sid, i, x)$.

The Coin Tossing Functionality $\mathcal{F}_{\mathsf{ct}}^{n}$:
Upon receiving (toss, sid) from all the honest parties, sample $x \leftarrow^{\$} [n]$ and broadcast $(\mathsf{tossed}, sid, x)$

Fig. 1. Functionalities $\mathcal{F}_{\mathsf{zk}}^{\mathcal{R}}$ and $\mathcal{F}_{\mathsf{ct}}$.

In order to achieve both efficient constructions and a high security level we will assume parties can safely erase information used in the protocol. This model was previously used in [Lin09] to show that MPC based on garbled circuits can achieve adaptive security, and more recently in Proof of Stack Blockchain constructions (e.g.[DGKR18]). The interested reader is referred to [Lin09] for a more detailed discussion about the practicality of secure erasures.

2.4 UC-SSLE definition

In this section we revise the definition of UC-SSLE through the SSLE functionality introduced in [CFG21] and extend it to also capture adaptive corruptions. We assume implicitly in the following definition that $\mathcal{F}_{\mathsf{SSLE}}$ is executed among a set of N users $P_1, \dots, P_N$. At a high level the first

The SSLE functionality $\mathcal{F}_{\mathsf{SSLE}}$:
Initialise $E, R \leftarrow \varnothing$, $n \leftarrow 0$ and call $\mathcal{Z}$ the environment. Upon receiving:

- $(\mathsf{register})$ from P_i: add $R \leftarrow R \cup \{(i, n)\}$, broadcast $(\mathsf{registered}, i)$ and set $n \leftarrow n + 1$.
- (elect, eid) from all honest parties: if $R \neq \varnothing$ and eid was not requested before sample $(j, \delta) \leftarrow^{\$} R$ and send $(\mathsf{outcome}, eid, 1)$ to P_j and $(\mathsf{outcome}, eid, 0)$ to P_i for $(i, \cdot) \in R$, $i \neq j$. Store $E \leftarrow E \cup \{(eid, j)\}$.
- (reveal, eid) from P_i: if $(eid, i) \in E$ broadcast $(\mathsf{result}, eid, i)$. Otherwise broadcast $(\mathsf{rejected}, eid, i)$.
- $(\mathsf{fake\_rejected}, eid, j)$ from $\mathcal{Z}$: If P_j is corrupted broadcast $(\mathsf{rejected}, eid, j)$.
- $(\mathsf{corrupt}, j)$ from $\mathcal{Z}$: Set $E_j = \{eid : (eid, j) \in E\}$ and reply with $(\mathsf{corrupted}, j, E_j)$

Fig. 2. Description of the SSLE functionality as in [CFG21] with active corruptions.

command $(\mathsf{register})$ allows a user P_j to be eligible in future election by adding a "ticket" (j, n) in R. Notice that there is no a priori bound on the number of times a user can register. Next, the command (elect, eid) samples a random ticket (j, δ) in R and secretly communicates the result to each party P, only revealing if P won by sending $(\mathsf{outcome}, eid, 0)$ or lost by sending $(\mathsf{outcome}, eid, 1)$. Note that by registering multiple times this mechanism allows to modify the probability in which users are elected, instead of selecting a uniformly random one. Finally the command (reveal, eid), sent by a user P_j who actually won election eid, makes the functionality broadcast a message $(\mathsf{result}, eid, j)$ that certifies his victory. Conversely, any attempt of non-winning users to claim victory through $\mathcal{F}_{\mathsf{SSLE}}$ results in the functionality broadcasting $(\mathsf{rejected}, eid, j)$.

Finally fake_rejected and corrupt are reserved to the adversary $\mathcal{Z}$. As pointed out in [CFG21], the first one is necessary for technical reasons: it captures the possibility that a winning malicious user P_j claims victory incorrectly by sending, for instance, some random message. Indeed, note that in such a case, the simulator would have no way to make $\mathcal{F}_{\mathsf{SSLE}}$ reject this incorrect claim and broadcast $(\mathsf{rejected}, eid, j)$ without this command. The second one is used when adversary corrupts a user P_j to let the former know the IDs of the elections won by P_j in the past. This models the fact that (a corrupted) P_j knows the set of elections in which she won.

3 Statically secure SSLE from DDH

3.1 Intuition and relations with previous work

We start by quickly revising the shuffle-based solutions from [BEHG20]. At a high level users maintain a list of n commitments (recall that n denotes the number of registered users) $c_{s,\ell} = (g_{s,\ell}, h_{s,\ell}) \in \mathbb{G}^2$. When a party registers she first commits to some $x \sim U(\mathbb{F}_q)$ by querying the RO on a random point $k \sim U(\{0,1\}^\lambda)$, setting $\mathcal{H}(k) = x||y$ and publishing y as a commitment to x. Next he appends to the list an ElGamal commitment $c_{s,n} = (g^r, g^{rx})$ and perform a shuffle, which is carried out by sampling a permutation $\eta : [n] \to [n]$ and setting $c_{s+1,\ell} = c_{s,\eta(\ell)}^{r_\ell}$ for $r_\ell \xleftarrow{\$} \mathbb{F}_q$. In order to elect the next leader, an index $\gamma \in [n]$ is chosen by the random beacon and the winner is whoever can provide x, k such that $c_{s,\gamma} = (u, u^x)$ and $\mathcal{H}(k) = x||y$ for some commitment y associated to them. Notice that when a user reveals that she won, her commitment is removed from the list and so she may have to register again. On top of that, to ensure no replication attack occurs, users need to check that the elements y provided at registration time are all different and that each new commitment c provided by another user is not in the form (u, u^x) for some secret key x they previously used. Finally, according to how the correctness of the shuffle is verified we distinguish two possible variants of the protocol sketched above:

- *Proof based version*: The shuffler provides a NIZK proof that the shuffle was performed correctly
- *Complaint based version*: No proof is provided. Each user checks that for each secret key x they have, the new list includes one and only one commitment of the form $c = (u, u^x)$. If that is not the case they reveal x and abort.

Surprisingly, we show in Appendix A.2 that the complaint based variant can satisfy game-based security (as defined in [BEHG20]) *only if* when a complaint occurs the protocol is aborted and concluded, meaning that no more elections can be performed afterwards (even if a new setup is executed). If this condition is not satisfied, we prove that the protocol is actually insecure, see appendix A.2 for details.

As a consequence, the only viable option to achieve UC security is to build on the proof based variant. Towards this goal, a difficulty arises from the fact that, because of shuffling, $\mathcal{S}$ has no way to identify commitments belonging to malicious parties and this seems necessary to simulate the right index γ when $\mathcal{F}_{\mathsf{SSLE}}$ signals that a corrupted P_j won an election. A way to address this is to employ a UC-NIZK to argue correctness of the shuffle. This would allow $\mathcal{S}$ to extract the permutation used, but it would also induce a significant overhead. We follow a different and much simpler path: instead of the commitment y (generated through the random oracle), each party provides $H = g^x$ together with a (compact and easy to realize) UC-NIZK that she knows the secret x used in the commitment. In this way $\mathcal{S}$, knowing all the secret keys, can link each commitment to its issuer.

Finally, we notice that another advantage of our protocol, with respect to the original one, is that it reduces by half the overall communication costs. Indeed, rather than keeping $(g^{r_\ell}, g^{r_\ell x_\ell})_{\ell \in [n]}$ in the list we simply store $(g^r, g^{rx_1}, \dots, g^{rx_n})$.

3.2 Construction secure against static corruptions

Here we describe the UC-secure against static corruptions construction detailed in Figure 3. To clarify notation we denote with R a list that links user's ID to the commitment they provide at registration time, $\mathcal{K}$ a *keyring*, i.e. a set of secret keys belonging to P_i, n the number of registrations performed so far and s the amount of shuffles executed. Next we describe the protocol in Fig. 3 breaking it down to the following phases:

- *Registration Phase*: In lines 1-4, P_i creates a new secret key x, two commitments to it $H = g^x$ and $h_{s,n} = g_s^x$ – where the second one is later appended to the maintained list of commitments – and a UC-NIZK of discrete logarithmic equality. Finally P_i performs a shuffle (see below). Parties accepts the registration if the UC-NIZK is correct, lines 9-11, modelled by a proof message from $\mathcal{F}_{\mathsf{zk}}^{\mathcal{R}_{\mathsf{DH}}}$.
- *Shuffle phase*: In lines 5-8, P_i samples randomness $r \in \mathbb{F}_q$ and $\eta : [n] \to [n]$ a permutation and shuffle the previously given vector $g_s, h_{s,1}, \dots, h_{s,n}$ by setting $h_{s+1,\ell} \leftarrow h_{s,\eta(\ell)}^r$ and adding a NIZK to ensure correctness. Other parties accepts the shuffle, lines 12-13 only if the proof is valid.
- *Election phase*: To elect a leader, lines 14-16, users query the random beacon $\mathcal{F}_{\mathsf{ct}}^n$ which returns $\gamma \in [n]$. Each user then checks if any of her stored keys x is such that $h_{s,\gamma} = g_s^x$, in which case she is the winner.
- *Reveal phase*: If P_i won election *eid*, meaning that he knows an x such that $h_{s,n} = g_s^x$, in line 17-22 she claims victory by recovering the commitment H to x provided at registration phase and by proving that H in base g and $h_{s,n}$ in base g_s have the same discrete logarithm. Other parties in lines 23-25 accept the claim only if the proof is correct and H was associated to P_i.

As a final note, we remark that both our protocols (i.e. the one presented in this section and the adaptively secure one given in the next section) manage to achieve the full UC-security notion defined in [CFG21] without needing to resort to their parametrized variant. In appendix B.2 we prove the following theorem.

Theorem 1. *If the Decisional Diffie-Hellman Problem is hard in $\mathbb{G}$, Protocol 3 UC-realises $\mathcal{F}_{\mathsf{SSLE}}$ in the $(\mathcal{F}_{\mathsf{ct}}, \mathcal{F}_{\mathsf{zk}}^{\mathcal{R}})$-hybrid model against unbounded static corruptions.*

4 Adaptively secure SSLE with Erasures from DDH

4.1 Intuition

In order to obtain a protocol that UC-realises $\mathcal{F}_{\mathsf{SSLE}}$ against adaptive corruptions, we will now discuss how to modify Protocol 3. The major issue there arises when $\mathcal{Z}$ corrupts a user P_i after several elections took place as this implies that the simulator $\mathcal{S}$ has to produce an exponent x such that $h_{s,\ell} = g_s^x$ for each step s and some $\ell \in [n]$ consistently with the outcome of previous elections. The trouble is that this exponent may well not exists since each time an honest user wins $\mathcal{F}_{\mathsf{SSLE}}$ does

Party P_i realising $\mathcal{F}_{\mathsf{SSLE}}$:

```
    Initially set $R, \mathcal{K} \leftarrow \varnothing,\ n, s \leftarrow 0,\ g_0 \leftarrow g$. On input
1 : $(\mathsf{register})$: Get $x \xleftarrow{\$} \mathbb{F}_q$, $H \leftarrow g^x$ and store the key $\mathcal{K} \leftarrow \mathcal{K} \cup \{x\}$
2 :   Set $h_{s,n} \leftarrow g_s^x$ the new commitment to append in the list
3 :   Prove knowledge of $x$ sending $(\mathsf{prove}, n, (g, g_s), (H, h_{s,n}), x)$ to $\mathcal{F}_{\mathsf{zk}}^{\mathcal{R}_{\mathsf{DH}}}$
4 :   Update $n \leftarrow n + 1$
        // Shuffle:
5 :   Sample $r \xleftarrow{\$} \mathbb{F}_q$ and a permutation $\eta \xleftarrow{\$} S_n$
6 :   Compute $g_{s+1} \leftarrow g_s^r$ and for all $\ell \in [n]$ set $h_{s+1,\ell} \leftarrow h_{s,\eta(\ell)}^r$.
7 :   Prove shuffle correctness $\pi \leftarrow \mathsf{NIZK.P}_{\mathsf{sh}}(g_s, \mathbf{h}_s, g_{s+1}, \mathbf{h}_{s+1}, (r, \eta))$
8 :   Broadcast the shuffled list $(\mathsf{shuffle}, n, g_{s+1}, \mathbf{h}_{s+1}, \pi)$
9 : $(\mathsf{proof}, n, j, (g, H))$ from $\mathcal{F}_{\mathsf{zk}}^{\mathcal{R}_{\mathsf{DH}}}$ with $(\cdot, \cdot, H) \notin R$:
10 :  Link $H$ to $P_j$ adding $R \leftarrow R \cup \{(j, n, H)\}$.
11 :  Set $n \leftarrow |R|$ and return $(\mathsf{registered}, j)$
12 : $(\mathsf{shuffle}, sid, g_{s+1}, \mathbf{h}_{s+1}, \pi)$ from $P_j$:
13 :  If the proof $\pi$ is accepted, update $s \leftarrow s + 1$

14 : $(\mathsf{elect}, eid)$: Send $(\mathsf{toss}, eid)$ to $\mathcal{F}_{\mathsf{ct}}^n$ and wait for $(\mathsf{tossed}, eid, \gamma)$
15 :  If $g_s^x = h_{s,\gamma}$ for some key $x \in \mathcal{K}$: Return $(\mathsf{outcome}, eid, 1)$
16 :  Else return $(\mathsf{outcome}, eid, 0)$

17 : $(\mathsf{reveal}, eid)$:
18 :  If $P_i$ won election $eid$, i.e. if $g_s^x = h_{s,\gamma}$ for an $x \in \mathcal{K}$:
19 :    Find $(i, \delta, H) \in R$ such that $H = g^x$
20 :    Prove a DL equality relation $\pi \leftarrow \mathsf{NIZK.P}_{\mathsf{DH}}((g, g_s), (H, h_{s,\gamma}), x)$
21 :    Broadcast $(\mathsf{claim}, eid, \delta, \pi)$ and execute a shuffle, lines 5-8.
22 :  Else broadcast an error message $(\mathsf{claim}, eid, \perp)$.
23 : $(\mathsf{claim}, eid, \delta, \pi)$ from $P_j$:
24 :  If $(j, \delta, H) \in R$ and $\pi$ is accepted: Return $(\mathsf{result}, eid, j)$
25 :  Else: Return $(\mathsf{rejected}, eid, j)$
```

Fig. 3. UC-SSLE from DDH, secure against static corruption.

not immediately reveal its identity, forcing $\mathcal{S}$ to point through $\mathcal{F}_{\mathsf{ct}}^{n}$ a random commitment "linked" to some honest user.

While more sophisticated primitives, like non-committing encryption, could be adapted to address the issue, we take a different path by assuming parties can perform secure data erasures. We design a key update phase occurring after each shuffle[5] that untangle the honest users' secret keys from commitments sent in previous steps. To make this idea work, we need an equivocal commitment in order to correctly simulate at least the last election, and the simplest choice is Pedersen with base $\mathbf{g}_s \in \mathbb{G}^2$. Thus, at registration time we make P_i send $h_{s,n} = \mathbf{g}_s^{(\alpha,n)}$, store the key (α, δ) and let the index n becomes publicly associated to P_i, which will be the only party capable of opening a given commitment to n.

It remains to discuss how to perform key update. Ideally, P_j could update a secret key (α, δ) by sampling $\omega \leftarrow^{\$} \mathbb{F}_q$ and sending $u = \mathbf{g}_s^{(\omega,0)}$. Such an u could then be combined to P_j's commitment $h_{s,\ell} = \mathbf{g}_s^{(\alpha,\delta)}$ so to obtain the update key as follows

$$h_{s,\ell} \cdot u = \mathbf{g}_s^{(\alpha,\delta)} \cdot \mathbf{g}_s^{(\omega,0)} = \mathbf{g}_s^{(\alpha+\omega,\delta)}.$$

Hence P_j could store $(\alpha + \omega, \delta)$ as the new key and erase the old one.

This simple solution does not work as, because of shuffling, there is no way to link P_j to her corresponding $h_{s,\ell}$. This means that parties (other than P_j) have no way to determine which element in the list has to be multiplied by the u sent by P_j. We fix this by keeping *two* different lists of n elements each: $(h_{s,\ell})_{\ell \in [n]}$ whose entries are shuffled and $(k_{s,\delta})_{\delta \in [n]}$ whose entries are never shuffled. At any step each party P_j, for each key (α, δ) she knows, is linked to a commitment $h_{s,\ell}$ if $\mathbf{g}_s^{(\alpha,\delta)} = h_{s,\ell} \cdot k_{s,\delta}$. In this way updates can be applied to $k_{s,\delta}$, which is publicly associated to P_j, even though the second component $h_{s,\ell}$ is not.

4.2 Construction secure against active corruptions

We detail a protocol that applies all these ideas in Figure 4. As before, below we provide a step-by-step explanation of each phase:

- *Registration phase*: In lines 1-3 P_i creates a Pedersen commitment of n, which is the smallest index not yet used, by sampling $\alpha \leftarrow^{\$} \mathbb{F}_q$ and setting $h_{s,n} = \mathbf{g}_s^{(\alpha,n)}$. Next it stores the new key (α, n) in a keyring $\mathcal{K}$, proves knowledge of α through $\mathcal{F}_{\mathsf{zk}}^{\mathcal{R}_{\mathsf{ped}}}$ and performs a shuffle, see below. Once other users receive the proof, formally modelled as a proof command from $\mathcal{F}_{\mathsf{zk}}^{\mathcal{R}_{\mathsf{ped}}}$, in lines 8-9 they record in R that n is associated to P_i and increase n. Notice P_i will be the only user linked to n.
- *Shuffle phase*: In lines 4-7 P_i samples a field element $r \in \mathbb{F}_q$ and a permutation $\eta : [n] \to [n]$. With them it exponentiates all elements to the power of r and shuffles only entries of the list $(h_{s,\ell})_{\ell \in [n]}$. The resulting list is sent along with a proof of correctness π.
- *Update phase*: Upon receiving a correct shuffle from another user, lines 10-16, P_j updates its keys stored in $\mathcal{K}$. For each of them (α, δ) it samples $\omega \in \mathbb{F}_q$ and create $u_{s+1,\delta} = \mathbf{g}_s^{(\omega,0)}$ a commitment to 0 together with a proof. Once other users receives the update, 17-18, if the proof is correct they multiply $k_{s,\delta}$ by the new commitment. Notice that in this way $k_{s,\delta}$ is always a Pedersen commitment to zero.

[5] In Section 4.3 we show how to perform it less frequently

- *Election phase*: All users query the random beacon $\mathcal{F}_{\mathsf{ct}}^{n}$, lines 19-22, which returns $\gamma \in [n]$. Each user then checks if she can open $h_{s,\gamma} \cdot k_{s,\delta}$ to (α, δ) for some (α, δ) in her keyring, in which case she is the winner.
- *Reveal phase*: When P_i won an election and wish to reveal it, lines 23-26, it prove through a NIZK that he is able to open $h_{s,\gamma} \cdot k_{s,\delta}$ to δ, for some δ publicly linked to him. Parties who receive the claim, lines 27-29, accept it only if δ is associated to P_i and the proof is correct.

In appendix B.3 we prove the following result.

Theorem 2. *If the Decisional Diffie-Hellman Problem is hard in $\mathbb{G}$, Protocol 4 UC-realises $\mathcal{F}_{\mathsf{SSLE}}$ in the $(\mathcal{F}_{\mathsf{ct}}, \mathcal{F}_{\mathsf{zk}}^{\mathcal{R}})$-hybrid model against unbounded active corruptions.*

4.3 Practical considerations

We now provide a series of optimisations and trade-off that were not included in Protocol 4 to keep the exposition simple. First we show how to reduce the frequency of updates, which represents the most expensive step in our solution both in terms of communication and in the number of parties who should actively take part. Next we also present a way to reduce the update cost by providing smaller NIZKs

- *Update only before an election*: since the goal of the update phase is to erase secret information linked to previous elections, there is no point in updating the key if several shuffles are taking place due to new registrations. Hence it suffices to update keys just after those shuffles that occur right before an election.
- *Update every ν elections*: The Pedersen commitment used $h = \mathbf{g}^{(\alpha,\delta)}$ needs to have randomness $\alpha \in \mathbb{F}_q$ to allow the simulator to equivocates results in the last election in case of corruption. A natural generalisation would be to maintain a larger base $\mathbf{g} \in \mathbb{G}^{\nu+1}$ and to commit using ν random field elements $\alpha_1, \dots, \alpha_\nu \in \mathbb{F}_q$ setting $h = \mathbf{g}^{(\alpha_1,\dots,\alpha_\nu,\delta)}$. In this way a simulator can equivocate the result of the previous ν elections, effectively reducing the need to perform a key update once every ν.
- *Smaller NIZK*: A downside of the previous point is that users need to prove knowledge of a secret key $\alpha_1, \dots, \alpha_\nu \in \mathbb{F}_q^\nu$ such that $h = \mathbf{g}^{(\alpha_1,\dots,\alpha_\nu,\delta)}$ at registration time and that $u = \mathbf{g}^{(\omega_1,\dots,\omega_\nu,0)}$ during an update. Using directly generalised Schnorr proofs [Mau15] would lead to arguments of size $O(\nu)$.
  A more efficient choice is to use Bulletproof's inner product argument [BBB$^+$18] made non-interactive through the Fiat-Shamir transform. This requires only $\Theta(\log \nu)$ group elements per proof. Note that the non-interactive version of Bulletproof was recently proved to be an argument of knowledge in [AFK21, Wik21].

5 Comparisons

In this section we compare our two constructions in terms of communication complexity with the shuffle based one in [BEHG20] and the one based on functional encryption [CFG21]. More specifically in Figure 5 we report the cumulative cost of performing up to 200 elections among $N = 2^{14}$ users[6] interleaving between each two a fixed amount of registrations, as done in [CFG21].

[6] This number of users was originally suggested in [Lab19]

Party P_i realising $\mathcal{F}_{\mathsf{SSLE}}$:

```
    Initialize $R, \mathcal{K} \leftarrow \varnothing$ and $n, s \leftarrow 0$. Trough $\mathcal{F}_{\mathsf{ct}}$ sample a random $\mathbf{g}_0 \in \mathbb{G}^2$. On input
1 : (register): Sample a new key $\alpha \xleftarrow{\$} \mathbb{F}_q$ and store it in the keyring $\mathcal{K} \leftarrow \mathcal{K} \cup \{(\alpha, n)\}$
2 :   Set $\mathbf{x} \leftarrow (\alpha, n),\ k_{s,n} \leftarrow 1$ and compute the commitment $h_{s,n} \leftarrow \mathbf{g}_s^{\mathbf{x}}$,
3 :   Send $(\mathsf{prove}, n, (\mathbf{g}_s, h_{s,n}, n), \alpha)$ to $\mathcal{F}_{\mathsf{zk}}^{\mathcal{R}_{\mathsf{ped}}}$ and set $n \leftarrow n + 1$
        // Shuffle:
4 :   Sample $r \xleftarrow{\$} \mathbb{F}_q$ and a permutation $\eta \xleftarrow{\$} S_n$
5 :   Shuffle by setting $\mathbf{g}_{s+1} \leftarrow \mathbf{g}_s^r,\ h_{s+1,j} \leftarrow h_{s,\eta(j)}^r,\ k_{s+1,j} \leftarrow k_{s,j}^r$
6 :   Prove correctness $\pi \leftarrow \mathsf{NIZK.P}_{\mathsf{Esh}}(\mathbf{g}_s, \mathbf{h}_s, \mathbf{k}_s, \mathbf{g}_{s+1}, \mathbf{h}_{s+1}, \mathbf{k}_{s+1}, (r, \eta))$
7 :   Erase $(r, \eta)$ and broadcast the shuffled list $(\mathsf{shuffle}, n, \mathbf{g}_{s+1}, \mathbf{h}_{s+1}, \mathbf{k}_{s+1}, \pi)$:
8 : $(\mathsf{proof}, j, (\mathbf{g}_s, h_{s,n}, n))$ from $\mathcal{F}_{\mathsf{zk}}^{\mathcal{R}_{\mathsf{ped}}}$:
9 :   Link $n$ to $P_j$ storing $R \leftarrow R \cup \{(j, n)\}$. Update $n \leftarrow |R|$ and return $(\mathsf{registered}, j)$
10 : $(\mathsf{shuffle}, sid, \mathbf{g}_{s+1}, \mathbf{h}_{s+1}, \mathbf{k}_{s+1}, \pi)$ from $P_j$ with accepting $\pi$:
11 :   For all previously stored keys $(\alpha, \delta) \in \mathcal{K}$:
12 :     Sample $\omega \xleftarrow{\$} \mathbb{F}_q$ and compute the update element $\mathbf{w} \leftarrow (\omega, 0),\ u_{s+1,\delta} \leftarrow \mathbf{g}_{s+1}^{\mathbf{w}}$
13 :     Prove knowledge of $\omega$ setting $\pi \leftarrow \mathsf{NIZK.P}_{\mathsf{ped}}((\mathbf{g}_{s+1}, u_{s+1,\delta}, 0), \omega)$
14 :     Update the old key $\alpha \leftarrow \alpha + \omega$ and erase the new term $\omega$
15 :     Broadcast the update element $(\mathsf{update}, sid, u_{s+1,\delta}, \pi)$
16 :   Update $s \leftarrow s + 1$
17 : $(\mathsf{update}, sid, u_{s,\delta}, \pi)$ from $P_j$:
18 :   If $\pi$ is accepted and $(j, \delta) \in R$: Update $k_{s,\delta} \leftarrow k_{s,\delta} \cdot u_{s,\delta}$
19 : $(\mathsf{elect}, eid)$: Send $(\mathsf{toss}, eid)$ to $\mathcal{F}_{\mathsf{ct}}^n$ and wait for $(\mathsf{tossed}, eid, \gamma)$
20 :   If some key $(\alpha, \delta) \in \mathcal{K}$ opens the commitment $h_{s,\gamma} \cdot k_{s,\delta}$, i.e. if $\mathbf{g}_s^{(\alpha,\delta)} = h_{s,\gamma} \cdot k_{s,\delta}$:
21 :     Return $(\mathsf{outcome}, eid, 1)$
22 :   Else: Return $(\mathsf{outcome}, eid, 0)$
23 : $(\mathsf{reveal}, eid)$: If $P_i$ won election $eid$, i.e. if for some $(\alpha, \delta) \in \mathcal{K}$, $\mathbf{g}_s^{(\alpha,\delta)} = h_{s,\gamma} \cdot k_{s,\delta}$:
24 :     Prove knowledge of $\alpha$ by setting $\pi \leftarrow \mathsf{NIZK.P}_{\mathsf{ped}}((\mathbf{g}_s, h_{s,\gamma} \cdot k_{s,\delta}, \delta), \alpha)$
25 :     Broadcast $(\mathsf{claim}, eid, \delta, \pi)$ and execute a shuffle, lines 4-7
26 :   Else: Broadcast $(\mathsf{claim}, eid, \perp)$
27 : $(\mathsf{claim}, eid, \delta, \pi)$ from $P_j$:
28 :   If $\pi$ is accepted and $(j, \delta) \in R$: Accept $P_j$ as the leader returning $(\mathsf{result}, eid, j)$
29 :   Else: Return $(\mathsf{rejected}, eid, j)$
```

Fig. 4. UC-SSLE from DDH with erasures secure against active corruptions.

In light of Section 4.3 we set $\nu = \Theta(\sqrt{N \log N})$ to equate the amortised cost coming from the update phase and the longer base $\mathbf{g} \in \mathbb{G}^{\nu+1}$ maintained at each shuffle. To instantiate the NIZK proof of correct shuffling we applied [BG12] with proof size $O(\sqrt{N})$. While more succinct arguments (e.g. [CHM+20, MBKM19]) providing proofs of constant size are known, translating the shuffle relation into the NP-complete problem solved by these argument systems would significantly affect the prover's time.

With this choice of parameters we observe that the cost of a single shuffle in Protocol 3 amounts to 567KB for 2^{14} users, almost half of the 1.10MB required in [BEHG20]. Our second construction, Protocol 4, instead achieves comparable efficiency with the previous construction requiring only 13.1KB more per shuffle, yet guaranteeing a significantly higher level of security. Regarding [CFG21] instead their solution does not rely on shuffles and provides efficient registrations at the cost a setup phase high in communication that has to be performed every ~ 200 rounds. For this reason our construction, as [BEHG20], performs better when less registrations are performed per round.

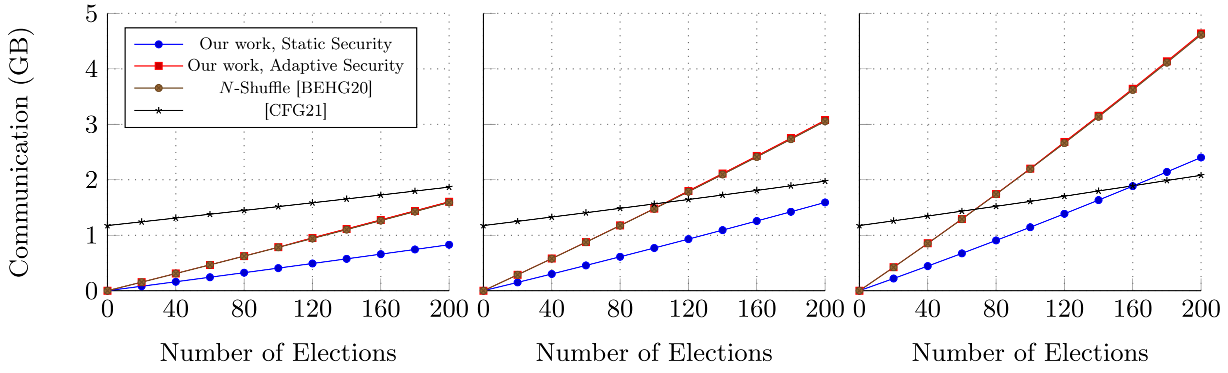

Fig. 5. Cumulative communication complexity in our statically secure construction, Figure 3, adaptively secure, Figure 4, the shuffle based [BEHG20] with one ballot of size N and in [CFG21]. Protocol starts with $N = 2^{14}$ users and between every two elections 6 (left graph), 12 (middle graph) or 18 (right graph) registrations occur. Note that after ~ 200 elections [CFG21] requires a new setup.

Acknowledgements

This work has received funding in part from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program under project PICOCRYPT (grant agreement No. 101001283), by the Spanish Government under projects SCUM (ref. RTI2018-102043-B-I00), RED2018-102321-T, and SECURING (ref. PID2019-110873RJ-I00), by the Madrid Regional Government under project BLOQUES (ref. S2018/TCS-4339), by a research grant from Nomadic Labs and the Tezos Foundation, by the Programma ricerca di ateneo UNICT 35 2020-22 linea 2 and by research gifts from Protocol Labs.

References

ABC+05. Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 205–222. Springer, Heidelberg, August 2005.

AC21. Sarah Azouvi and Daniele Cappelletti. Private attacks in longest chain proof-of-stake protocols with single secret leader elections. In *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*, pages 170–182, 2021.

AFK21. Thomas Attema, Serge Fehr, and Michael Klooß. Fiat-shamir transformation of multi-round interactive proofs. Cryptology ePrint Archive, Report 2021/1377, 2021. `https://eprint.iacr.org/2021/1377`.

AMM18. Sarah Azouvi, Patrick McCorry, and Sarah Meiklejohn. Betting on blockchain consensus with fantomette. *CoRR*, abs/1805.06786, 2018.

BBB$^+$18. Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Greg Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy*, pages 315–334. IEEE Computer Society Press, May 2018.

BDOP04. Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 506–522. Springer, Heidelberg, May 2004.

BEHG20. Dan Boneh, Saba Eskandarian, Lucjan Hanzlik, and Nicola Greco. Single secret leader election. In *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*, pages 12–24, 2020.

BG12. Stephanie Bayer and Jens Groth. Efficient zero-knowledge argument for correctness of a shuffle. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 263–280. Springer, Heidelberg, April 2012.

BGG$^+$18. Dan Boneh, Rosario Gennaro, Steven Goldfeder, Aayush Jain, Sam Kim, Peter M. R. Rasmussen, and Amit Sahai. Threshold cryptosystems from threshold fully homomorphic encryption. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 565–596. Springer, Heidelberg, August 2018.

BGI$^+$01. Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 1–18. Springer, Heidelberg, August 2001.

BGM16. Iddo Bentov, Ariel Gabizon, and Alex Mizrahi. Cryptocurrencies without proof of work. In Jeremy Clark, Sarah Meiklejohn, Peter Y. A. Ryan, Dan S. Wallach, Michael Brenner, and Kurt Rohloff, editors, *FC 2016 Workshops*, volume 9604 of *LNCS*, pages 142–157. Springer, Heidelberg, February 2016.

BPS16. Iddo Bentov, Rafael Pass, and Elaine Shi. Snow white: Provably secure proofs of stake. Cryptology ePrint Archive, Report 2016/919, 2016. `https://eprint.iacr.org/2016/919`.

BSW11. Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, March 2011.

Can01. Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

CD20. Ignacio Cascudo and Bernardo David. ALBATROSS: Publicly AttestabLe BATched Randomness based On Secret Sharing. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 311–341. Springer, Heidelberg, December 2020.

CDG$^+$18. Jan Camenisch, Manu Drijvers, Tommaso Gagliardoni, Anja Lehmann, and Gregory Neven. The wonderful world of global random oracles. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 280–312. Springer, Heidelberg, April / May 2018.

CF01. Ran Canetti and Marc Fischlin. Universally composable commitments. In Joe Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 19–40. Springer, Heidelberg, August 2001.

CFG21. Dario Catalano, Dario Fiore, and Emanuele Giunta. Efficient and universally composable single secret leader election from pairings. Cryptology ePrint Archive, Report 2021/344, 2021. `https://eprint.iacr.org/2021/344`.

CFG22. Dario Catalano, Dario Fiore, and Emanuele Giunta. Adaptively secure single secret leader election from ddh. Cryptology ePrint Archive, 2022.

CFGN96. Ran Canetti, Uriel Feige, Oded Goldreich, and Moni Naor. Adaptively secure multi-party computation. In *28th ACM STOC*, pages 639–648. ACM Press, May 1996.

CHM$^+$20. Alessandro Chiesa, Yuncong Hu, Mary Maller, Pratyush Mishra, Noah Vesely, and Nicholas P. Ward. Marlin: Preprocessing zkSNARKs with universal and updatable SRS. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 738–768. Springer, Heidelberg, May 2020.

CJS14. Ran Canetti, Abhishek Jain, and Alessandra Scafuro. Practical UC security with a global random oracle. In Gail-Joon Ahn, Moti Yung, and Ninghui Li, editors, *ACM CCS 2014*, pages 597–608. ACM Press, November 2014.

DGKR18. Bernardo David, Peter Gazi, Aggelos Kiayias, and Alexander Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part II*, volume 10821 of *LNCS*, pages 66–98. Springer, Heidelberg, April / May 2018.

DKT$^+$20. Amir Dembo, Sreeram Kannan, Ertem Nusret Tas, David Tse, Pramod Viswanath, Xuechao Wang, and Ofer Zeitouni. Everything is a race and nakamoto always wins. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 859–878. ACM Press, November 2020.

DNRS03. Cynthia Dwork, Moni Naor, Omer Reingold, and Larry Stockmeyer. Magic functions: In memoriam: Bernard m. dwork 1923–1998. *Journal of the ACM (JACM)*, 50(6):852–921, 2003.

FLS90. Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *31st FOCS*, pages 308–317. IEEE Computer Society Press, October 1990.

GGH$^+$13. Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

GHM$^+$17. Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, SOSP '17, page 51–68, New York, NY, USA, 2017. Association for Computing Machinery.

GOT19. Chaya Ganesh, Claudio Orlandi, and Daniel Tschudi. Proof-of-stake protocols for privacy-aware blockchains. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 690–719. Springer, Heidelberg, May 2019.

Hof11. Dennis Hofheinz. Possibility and impossibility results for selective decommitments. *Journal of Cryptology*, 24(3):470–516, July 2011.

KKKZ19. Thomas Kerber, Aggelos Kiayias, Markulf Kohlweiss, and Vassilis Zikas. Ouroboros crypsinous: Privacy-preserving proof-of-stake. In *2019 IEEE Symposium on Security and Privacy*, pages 157–174. IEEE Computer Society Press, May 2019.

KMTZ13. Jonathan Katz, Ueli Maurer, Björn Tackmann, and Vassilis Zikas. Universally composable synchronous computation. In Amit Sahai, editor, *TCC 2013*, volume 7785 of *LNCS*, pages 477–498. Springer, Heidelberg, March 2013.

Lab19. Protocol Labs. Secret single-leader election (SSLE). `https://web.archive.org/web/20191228170149/https://github.com/protocol/research-RFPs/blob/master/RFPs/rfp-6-SSLE.md`, 2019.

Lin09. Andrew Y. Lindell. Adaptively secure two-party computation with erasures. In Marc Fischlin, editor, *CT-RSA 2009*, volume 5473 of *LNCS*, pages 117–132. Springer, Heidelberg, April 2009.

Mau15. Ueli Maurer. Zero-knowledge proofs of knowledge for group homomorphisms. *Designs, Codes and Cryptography*, 77(2):663–676, 2015.

MBKM19. Mary Maller, Sean Bowe, Markulf Kohlweiss, and Sarah Meiklejohn. Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 2111–2128. ACM Press, November 2019.

Ped92. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 129–140. Springer, Heidelberg, August 1992.

Sah99. Amit Sahai. Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999.

Wik21. Douglas Wikström. Special soundness in the random oracle model. Cryptology ePrint Archive, Report 2021/1265, 2021. `https://eprint.iacr.org/2021/1265`.

A Attacks to the Complaint-based Construction

A.1 Game-based security definition

Before proving that the complaint-based protocol in [BEHG20] is insecure if resumed or restarted after an abort, we revise the game based security notion required to provide our attack. First we recall their definition of SSLE scheme.

Definition 1. *A Secret Single Leader Election scheme is a tuple of protocols* ($\mathsf{SSLE.Setup}$, $\mathsf{SSLE.Reg}$, $\mathsf{SSLE.Elect}$, $\mathsf{SSLE.Claim}$, $\mathsf{SSLE.Vrf}$) *executed among N users, such that*

- $\mathsf{SSLE.Setup}$ *returns public parameters* pp *to every player and* sp_i *to* P_i*.*
- $\mathsf{SSLE.Reg}_{\mathsf{pp}}(i)$ *registers player P_i for future elections*
- $\mathsf{SSLE.Elect}_{\mathsf{pp}}$ *returns publicly a challenge c.*
- $\mathsf{SSLE.Claim}_{\mathsf{pp}}(c, \mathsf{sp}_i, i) \to \pi / \perp$ *returns publicly a proof to claim victory.*
- $\mathsf{SSLE.Vrf}_{\mathsf{pp}}(c, \pi, i) \to 0/1$ *verifies the correctness of a claim.*

Among the three security notion introduced in [BEHG20], that are *uniqueness*, *fairness* and *unpredictability*, our attack will only address the latter which roughly states that before an honest winner reveal her victory, the adversary cannot guess her identity significantly better than at random. More in detail they introduce a security game as follows

The Unpredictability Experiment $\mathsf{Exp}^{\mathcal{A}}_{\mathsf{Unpr}}(1^\lambda, N)$:

```
1 : **When** $M \leftarrow \mathcal{A}(1^\lambda, N)$, simulate $P_i$ for $i \in [N] \setminus M$ interacting with $\mathcal{A}$
2 : Execute $\mathsf{SSLE.Setup} \to \mathsf{pp}, \mathsf{sp}_i$ for $i \in [N] \setminus M$. Set $s \leftarrow 0, R \leftarrow \varnothing$.
3 : **When** $\mathsf{register}, i \leftarrow \mathcal{A}$: Run $\mathsf{SSLE.Reg}_{\mathsf{pp}}(i)$ and add $R \leftarrow R \cup \{i\}$
4 : **When** $\mathsf{elect} \leftarrow \mathcal{A}$: Execute $\mathsf{SSLE.Elect}_{\mathsf{pp}} \to c_s$
5 :   $\pi_{i,s} \leftarrow \mathsf{SSLE.Claim}_{\mathsf{pp}}(c_s, \mathsf{sp}_i, i), \quad \mathcal{A} \leftarrow \pi_{i,s} \quad \forall i \in R \setminus M$
6 :   $\pi_{i,s} \leftarrow \mathcal{A} \quad \forall i \in R \cap M; \quad s \leftarrow s + 1$
7 : **When** $\mathsf{chall} \leftarrow \mathcal{A}$: Call $n = |R|$ and $\tau = |R \cap M|$;
8 :   Execute $\mathsf{SSLE.Elect}_{\mathsf{pp}} \to c_s$
9 :   Compute $\pi_{i,s} \leftarrow \mathsf{SSLE.Claim}_{\mathsf{pp}}(c, \mathsf{sp}_i, i) \quad \forall i \in R \setminus M$
10 :  Wait of the adversary to return $j \leftarrow \mathcal{A}$
11 :  Return 0 if any protocol fails or if $\mathsf{SSLE.Vrf}_{\mathsf{pp}}(c_s, \pi_{j,s}, j) \neq 1$, 1 otherwise
```

Fig. 6. Unpredictability experiment as defined in [BEHG20]

Definition 2. *An SSLE scheme satisfies unpredictability (against unbounded corruptions) if for every* $\mathsf{PPT}\mathcal{A}$ *there exists a negligible function ε such that*

$$\mathsf{Adv}(\mathcal{A}) := \Pr\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{Unpr}}(1^\lambda, N) = 1 \,\middle|\, \mathsf{HW}\right] - \frac{1}{n-\tau} \leq \varepsilon(\lambda)$$

where HW *is the event "$\exists i \in [N] \setminus M : \mathsf{SSLE.Vrf}(c_s, \pi_{i,s}, i) = 1$" requiring the existence of at least one honest winner in the challenge phase.*

Note that according to the following definition the advantage of $\mathcal{A}$ doesn't range in $[0, 1]$ as usual, but rather is bounded by

$$-\frac{1}{n-\tau} \leq \mathsf{Adv}(\mathcal{A}) \leq \frac{n-\tau-1}{n-\tau}.$$

A.2 Attack description

In this Section we illustrate a practical attack that could be mounted against the *complaint-based* construction in [BEHG20], briefly described in Section 3.1, if the protocol is not halted *and never resumed or restarted again* after a correct complain has been raised. Since the original protocol does not describe how to continue after a user performs a complain, we will study two natural ways of extending their construction:

- Π_1: Commitments associated to parties who complained are removed from the list.
- Π_2: All commitments in the list are removed, effectively restarting a new instance of the protocol.

We will now show both solutions fail to satisfy the unpredictability notion as defined in [BEHG20, CFG21]. Intuitively this happens because when a dishonest party performing a shuffle is caught cheating, he learns nothing whereas when he is not caught some information may be leaked. Combining this with a significant probability of not being caught and the possibility to amplify its chances of success through repetitions (something disallowed if, after cheating the first time, the protocol halts and never restart again) leads to successfully breaking unpredictability with close to 1 probability.

Proposition 1. *There exists a* PPT *algorithm $\mathcal{A}$ playing the Unpredictability game against Π_1 that performs at most 2λ registrations of honest users, λ registrations of malicious ones, and wins with advantage*

$$\mathsf{Adv}(\mathcal{A}) \; = \; \frac{1}{2}\left(1 - \frac{1}{2^\lambda}\right).$$

Proof. We provide a detailed description of $\mathcal{A}$ in Figure 7. Informally it repeats for at most λ times a cycle of three registrations, where the last one is of a malicious user who does not correctly shuffle, but instead produces a list obtained mixing previous ones.

Algorithm $\mathcal{A}$:

```
      Initially corrupt $P_2$ and set $\mathsf{cnt} \leftarrow 0$. Let $l$ be the list of commitments
 1 :  **While** $\mathsf{cnt} < \lambda$:
 2 :    Ask to register $P_0$ and parse $l = (c_0^0)$
 3 :    Ask to register $P_1$ and parse $l = (c_0^1, c_1^1)$
 4 :    Ask to register $P_2$:
 5 :    Sample a string $y \leftarrow^{\$} \{0,1\}^\lambda$, a commitment $c^* \leftarrow^{\$} \mathbb{G}^2$ and a bit $b \leftarrow^{\$} \{0,1\}$
 6 :    Send $y$ and the new list $l = (c_0^0, c_b^1, c^*)$
 7 :    **If** no party complains:
 8 :      **break**
 9 :    $\mathsf{cnt} \leftarrow \mathsf{cnt} + 1$
10 :  **If** $\mathsf{cnt} = \lambda$: Ask to register $P_0, P_1$ and $P_2$
11 :  Request a challenge election and wait for $\gamma \in [3]$ from the random beacon
12 :  **If** $\gamma \notin \{0,1\}$ abort; **Else** return $\gamma$.
```

Fig. 7. Description of $\mathcal{A}$ breaking unpredictability of Π_1

In order to prove the proposition we let for each cycle $b' \in \{0,1\}$ be such that P_1 knowns the secret key of $c^1_{b'}$ and let x_0, x_1 be the secret key owned respectively by P_0 and P_1. Then we have that

$$c^0_0 = (u, u^{x_0}), \qquad c^1_{b'} = (v, v^{x_1}), \quad c^1_{1-b'} = (w, w^{x_0})$$

for some $u, v, w \in \mathbb{G}$. To proceed we study how honest parties react after the registration and shuffle of P_2 according to two possible cases:

- $b \neq b'$: The first two commitments in the list are (u, u^{x_0}) and (w, w^{x_0}) meaning that P_1 will complain and reveal its key. P_0 will do the same since after the shuffle there are two commitment both using the same key x_0. Hence after this shuffle $l = \varnothing$.

- $b = b'$: The first two commitments in the list are (u, u^{x_0}) and (v, v^{x_1}) meaning that both P_0 and P_1 won't raise any complain. Notice that in this case the first commitment is linked to P_0 and the second one to P_1.

Next we observe that $b \neq b'$ occurs with probability $1/2$ since $b \sim U(\{0,1\})$ and in each repetition the coin b is independent from previous choices. Hence calling bad the event in which parties complain after each of the λ cycles, $\Pr[\mathsf{bad}] = 2^{-\lambda}$. Finally if bad occurs and $\gamma \in \{0,1\}$, that is if there is an honest winner in the challenge election, then P_γ wins with probability $1/2$, according to the permutation chosen by P_1 during its registration. Conversely if no user complained in any previous cycle, as pointed out before the first commitment is liked to P_0 and the second one to P_1, hence $\mathcal{A}$ guesses correctly. We conclude that, calling HW the event $\gamma \in \{0,1\}$

$$\begin{aligned}
\mathsf{Adv}(\mathcal{A}) &= \Pr[\mathcal{A}\mathsf{wins}|\mathsf{HW}] - \frac{1}{n-\tau} = \Pr[\mathcal{A}\mathsf{wins}|\mathsf{HW}] - \frac{1}{2} \\
&= \Pr[\mathcal{A}\mathsf{wins}|\mathsf{HW}, \neg\mathsf{bad}]\Pr[\neg\mathsf{bad}] + \Pr[\mathcal{A}\mathsf{wins}|\mathsf{HW}, \mathsf{bad}]\Pr[\mathsf{bad}] - \frac{1}{2} \\
&= 1 - \frac{1}{2^\lambda} + \frac{1}{2^{\lambda+1}} - \frac{1}{2} = \frac{1}{2}\left(1 - \frac{1}{2^\lambda}\right),
\end{aligned}$$

where $n = 3$ denotes the number of parties registered when the challenge election is requested and $\tau = 1$ the number of corrupted parties among them. The thesis follows.

Regarding the second version of the protocol Π_2 in which after a complain the state is cleared and the protocol is restarted, the exact same adversary breaks the unpredictability property

Proposition 2. *There exists a* PPT *algorithm* $\mathcal{A}$ *playing the Unpredictability game against* Π_2 *that performs at most* 2λ *registrations of honest users,* λ *registrations of malicious ones, and wins with advantage*

$$\mathsf{Adv}(\mathcal{A}) = \frac{1}{2}\left(1 - \frac{1}{2^\lambda}\right).$$

Proof. The algorithm $\mathcal{A}$ breaking security is identical to the one presented in the previous proposition, Figure 7. The argument is analogous to that in the proof of Proposition 1.

B Proofs

B.1 Preliminaries and Notation

Before presenting proofs for all theorems stated so far, we introduce the following auxiliary notation and results. First of all we generalise the notion of composition between two permutation, recalling that the substitution group is defined as $S_n = \{\sigma : [n] \to [n] \ : \ \sigma \text{ bijection}\}$. We naturally embed $\iota : S_n \to S_m$ with $n < m$ by mapping

$$\iota(\sigma) : [m] \to [m] \quad : \quad \iota(\sigma)(x) = \begin{cases} \sigma(x) & \text{If } x \in [n] \\ x & \text{If } x \in [m] \setminus [n] \end{cases}$$

i.e. by extending σ to be the identity over $[m] \setminus [n]$. With abuse of notation we can assume $S_n \subseteq S_m$, which allow us to compose bijection of different permutation groups.

Next we revise the notion of statistical distance between two random variables distributed over the same (finite and discrete) measure space

Definition 3. *Given a finite set S and two random variables $x, y \sim S$ we define their statistical distance as*

$$\Delta(x, y) = \frac{1}{2} \sum_{a \in S} |\Pr[x = a] - \Pr[y = a]|$$

If A is an event and $x \sim S$ a random variable we denote with $x_{|A}$ the conditional random variable such that for all $a \in A$, $\Pr[x_{|A} = a] = \Pr[x = a|A]$. The main result we will use later on in the proof of security is the following, which allows to bound the joint statistical distance of two vectors $(x_1, y_1), (x_2, y_2)$ using upper bounds on the distance of x_1, x_2 and of y_1, y_2 conditioned on $x_1 = x, x_2 = x$ for almost all x.

Proposition 3. *Given four random variables $x_1, x_2 \sim X$, $y_1, y_2 \sim Y$ and called $X^+ = \{a \in X : \Pr[x_i = a] > 0,\ i \in [2]\}$, if there exists $A \subseteq X$ such that*

$$P(x_1 \in A) \leq \varepsilon_1, \quad \Delta(x_1, x_2) \leq \varepsilon_2, \quad \Delta(y_{1|x_1=x}, y_{2|x_2=x}) \leq \varepsilon_3 \quad \forall x \in X^+ \setminus A,$$

for positive real numbers $\varepsilon_1, \varepsilon_2, \varepsilon_3 \in \mathbb{R}^+$, then $\Delta((x_1, y_1), (x_2, y_2)) \leq \varepsilon_1 + \varepsilon_2 + \varepsilon_3$.

B.2 Static Construction

Proof (of Theorem 1). The main challenge to face when providing a simulator $\mathcal{S}$ for this protocol interacting with $\mathcal{F}_{\mathsf{SSLE}}$ and an environment $\mathcal{Z}$ is to correctly simulate the election phase. In particular whenever a corrupted party wins, $\mathcal{S}$ will receive from $\mathcal{F}_{\mathsf{SSLE}}$ the winner's index – which will be the same returned by honest parties when eventually a reveal request is sent – therefore $\mathcal{S}$ has to return through $\mathcal{F}^n_{\mathsf{ct}}$ the right index $\gamma \in [n]$. Conversely when an honest party P_i wins $\mathcal{S}$ knows only that the winner is honest. Indeed $\mathcal{F}_{\mathsf{SSLE}}$ will reveal its identity only after $\mathcal{Z}$ has sent (reveal, eid) to $\mathcal{F}_{\mathsf{SSLE}}$ as P_i. Interestingly with significant probability the index γ returned by $\mathcal{S}$ will be wrong.

In the first case we use $\mathcal{F}^{\mathcal{R}_{\mathsf{DL}}}_{\mathsf{zk}}$ to extract at registration time secret keys belonging to corrupted users in order to correctly identity the right element $h_{s,\ell}$ at election time. In the second case instead we can solve the issue by simulating proofs contained in the claim. A detailed description of the simulator is presented in Figure 8.

Next we outline a sequence of hybrid functionalities to show that $\mathcal{S} \circ \mathcal{F}_{\mathsf{SSLE}}$ is indistinguishable from the real protocol:

Simulator $\mathcal{S}$:

```
Initially wait for $\mathcal{Z}$ to send $M \subseteq [N]$ the set of corrupted parties. Initialize $R \leftarrow \varnothing,\ n, s \leftarrow 0,\ g_0 \leftarrow g$ and $\varphi : [n] \to [N]$.
On input
1 : $(\mathsf{registered}, j)$ from $\mathcal{F}_{\mathsf{SSLE}}$: Sample $x_{s,n} \xleftarrow{\$} \mathbb{F}_q, H \xleftarrow{\$} \mathbb{G}$ and set $h_{s,n} \leftarrow g_s^{x_{s,n}}$.
2 :   Broadcast $(\mathsf{proof}, n, (g, g_s), (H, h_{s,n}))$
        // Shuffle:
3 :   Sample $r \xleftarrow{\$} \mathbb{F}_q$ and $\eta \xleftarrow{\$} S_n$
4 :   Compute $g_{s+1} \leftarrow g_s^r$, set $\varphi(n) = j$ and update $\varphi \leftarrow \varphi \circ \eta$
5 :   **For** all $\ell \in [n]$:
6 :     **If** $\varphi(\ell) \in M$: Compute $h_{s+1,\ell} = h_{s,\eta(\ell)}^r$ and call $x_{s+1,\ell} \leftarrow x_{s,\eta(\ell)}$
7 :     **Else**: Sample $x_{s+1,\ell} \leftarrow \mathbb{F}_q$ and compute $h_{s+1,\ell} \leftarrow g_{s+1}^{x_{s+1,\ell}}$
8 :   Simulate $\pi$ and broadcast $(\mathsf{shuffle}, sid, g_{s+1}, \mathbf{h}_{s+1}, \pi)$
9 : $(\mathsf{prove}, n, (g, g_s), (H, h_{s,n}), x)$ from $P_j$:
10 :   **If** $H = g^x$ and $h_{s,n} = g_s^x$:
11 :     Store $x_{s,n} \leftarrow x$, set $\varphi(n) = j$ and broadcast $(\mathsf{proof}, n, (g, g_s), (H, h_{s,n}))$
12 : $(\mathsf{shuffle}, sid, g_{s+1}, \mathbf{h}_{s+1}, \pi)$ from $P_j$:
13 :   **If** $\pi$ is accepted: Find $\eta \in S_n$ such that $h_{s+1,\ell} = g_{s+1}^{x_{s,\eta(\ell)}}$
14 :     Update $\varphi \leftarrow \varphi \circ \eta$.
15 : A request from $\mathcal{F}_{\mathsf{SSLE}}$ to send $(\mathsf{outcome}, eid)$:
16 :   **If** any corrupted $P_j$ will receive $(\mathsf{outcome}, eid, 1)$: Sample $\gamma \xleftarrow{\$} \varphi^{-1}(j)$
17 :   **Else**: Sample $\gamma \xleftarrow{\$} \varphi^{-1}([N] \setminus M)$
18 :   Broadcast $(\mathsf{tossed}, eid, \gamma)$ and let $\mathcal{F}_{\mathsf{SSLE}}$ send its messages
19 : $(\mathsf{result}, eid, j)$ from $\mathcal{F}_{\mathsf{SSLE}}$:
20 :   Sample $\gamma' \xleftarrow{\$} \varphi^{-1}(j)$, set $\delta \leftarrow \xi(\gamma')$, simulate $\pi$ and broadcast $(\mathsf{claim}, eid, \pi)$
21 : $(\mathsf{rejected}, eid, j)$ from $\mathcal{F}_{\mathsf{SSLE}}$:
22 :   Broadcast $(\mathsf{claim}, eid, \bot)$
23 : $(\mathsf{claim}, eid, \delta, \pi)$ from $P_j$:
24 :   If $(j, \delta, H) \in R$ for some $H$ and $\pi$ is accepted, send $(\mathsf{reveal}, eid)$ to $\mathcal{F}_{\mathsf{SSLE}}$ as $P_j$
```

Fig. 8. Description of simulator $\mathcal{S}$ executed with an environment $\mathcal{Z}$ interacting $\mathcal{S} \circ \mathcal{F}_{\mathsf{SSLE}}$

H_0: The real protocol

H_1: As the real protocol but every NIZK from honest users is produced running the simulator

H_2: As H_1 but initially set $\xi : [n] \to [n]$ and $\varphi : [n] \to [N]$. After a registration for P_j store $x_{s,n}$ the exponent such that $h_{s,n} = g_s^{s_{x,n}}$ and set $\varphi(n) = j$. After a shuffle performed by a corrupted shuffler find $\eta \in S_n$ such that $h_{s+1,\ell} = g_{s+1}^{x_{s+1,\eta(\ell)}}$, or, if the shuffler is honest, let η be the permutation used. Update $\varphi \leftarrow \varphi \circ \eta$, $\xi \leftarrow \xi \circ \eta$ and $x_{s+1,\ell} \leftarrow x_{s,\eta(\ell)}$

H_3: As H_2 but initially set $E \leftarrow \varnothing$. After any election in which $\mathcal{F}_{\mathsf{ct}}^n$ returned $(\mathsf{tossed}, eid, \gamma)$, set $j = \varphi(\gamma)$ and store $E \leftarrow E \cup \{(eid, j)\}$.

H_4: As H_3 but when a corrupted user P_j sends $(\mathsf{claim}, eid, \delta, \pi)$, honest users returns $(\mathsf{result}, eid, j)$ if $(j, \delta, \cdot) \in R$, π is accepted and $(eid, j) \in E$. Else they return $(\mathsf{rejected}, eid, j)$.

H_5: As H_4 but each honest user P_j upon receiving (reveal, eid) check if $(eid, j) \in E$. If this is the case broadcast $(\mathsf{claim}, eid, \xi(\gamma), \pi)$ with simulated π where $(\mathsf{tossed}, eid, \gamma)$ was returned by $\mathcal{F}_{\mathsf{ct}}^n$. Else, broadcast $(\mathsf{claim}, eid, \perp)$.

H_6: As H_5 but every time a honest user P_i perform a shuffle, for all $\ell \in [n]$ such that $\varphi(\ell) \notin M$ sample $x_{s+1,\ell} \xleftarrow{\$} \mathbb{F}_q$. Next for all $\ell \in [n]$ set $h_{s+1,\ell} \leftarrow g_{s+1}^{x_{s+1,\ell}}$.

H_7: As H_6 but for each election sample $(j, \cdot, \cdot) \xleftarrow{\$} R$ and add (eid, j) to E. Next sample $\gamma \xleftarrow{\$} \varphi^{-1}(j)$ and return $(\mathsf{tossed}, eid, \gamma)$.

H_8: As H_7 but for each election sample $(j, \cdot, \cdot) \xleftarrow{\$} R$, add (eid, j) to E and broadcast $(\mathsf{tossed}, eid, \gamma)$ with

$$\gamma \xleftarrow{\$} \begin{cases} \varphi^{-1}(j) & \text{If } j \in M \\ \varphi^{-1}([N] \setminus M) & \text{If } j \notin M \end{cases}.$$

Moreover when an honest P_j reveals sample $\gamma' \xleftarrow{\$} \varphi^{-1}(j)$ and $\delta \leftarrow \xi(j)$.

H_9: The simulated protocol $\mathcal{F}_{\mathsf{SSLE}} \circ \mathcal{S}$.

Next we argue that any pair of subsequent functionalities is indistinguishable against a PPT adversary. Trivial cases are $\mathsf{H}_0 \equiv \mathsf{H}_1$ form perfect HVZK, and $\mathsf{H}_1 \equiv \mathsf{H}_2 \equiv \mathsf{H}_3$ since beside computing ξ and E their behaviour is unaltered. The proof is completed by the following chain of claims.

Claim 1. *In H_2 for all s up to negligible probability $h_{s,\ell} = g_s^{x_{s,\ell}}$ and there exists a unique $\eta_{s+1} = \eta \in S_n$ such that $h_{s+1,\ell} = g_{s+1}^{x_{s,\eta(\ell)}}$.* We prove the statement by induction, where the base case $s = 0$ is trivial. Assuming the claim for s, if the next shuffler is honest by weak simulation extractability there exists a permutation $\mu \in S_n$ and $r \in \mathbb{F}_q$ such that $g_{s+1} = g_s^r$ and $h_{s+1,\ell} = h_{s,\mu(\ell)}^r$ which is equal to $g_s^{r x_{s,\mu(\ell)}} = g_{s+1}^{x_{s,\mu(\ell)}}$. Hence $\eta = \mu$ and is the only one since by construction all elements $h_{s,\ell}$ are different. The same argument applies when the next shuffle is honest. Finally since we define $x_{s+1,\ell} = x_{s,\eta(\ell)}$ we have that $h_{s+1,\ell} = g_{s+1}^{x_{s,\eta(\ell)}} = g_{s+1}^{x_{s+1,\ell}}$ which completes the proof.

Claim 2. *In H_2, for any step s and $\ell \in [n]$, $(\varphi(\ell), \xi(\ell), H) \in R$ and $H = g^{x_{s,\ell}}$.* Again we prove this by induction where the base case is trivially satisfied. Calling next ξ_s and φ_s the function ξ and φ at round s, assume the thesis to be true for s. If a user P_j registers sending $(\mathsf{prove}, n, (g, g_s), (H, h_{s,n}), x)$ by construction $\xi(n) = n$, $\varphi(n) = j$ and $(j, n, H) \in R$. Moreover $x_{s,n}$ is such that $H = g^{x_{s,\xi(n)}}$ as claimed. After the $(s+1)$-th shuffle let $r \in \mathbb{F}_q$ and $\eta \in \S_n$ be the witness he used, which exists either by simulation soundness if the shuffler is corrupted or by construction otherwise. Then $\xi_{s+1} = \xi_s \circ \eta$ and $\varphi_{s+1} = \varphi_s \circ \eta$ implies

$$\forall \ell : \quad (\varphi_s(\eta(\ell)), \xi_s(\eta(\ell)), H) \in R \quad \Rightarrow \quad (\varphi_{s+1}(\ell), \xi_{s+1}(\ell), H) \in R$$

where $H = g^{x_{s,\eta(\ell)}} = g^{x_{s+1,\ell}}$. The claim is therefore proven.

Claim 3. $\mathsf{H}_3 \equiv \mathsf{H}_4$: The behaviour of H_3 and H_4 differs only when a corrupted P_j sends a message $(\mathsf{claim}, eid, \delta, \pi)$ that would be accepted in H_3 but not in H_4 executed with the same random coins.

Since the message would be accepted in H_3 we have that $(j, \delta, H) \in R$ an there exists by simulation soundness up to negligible probability an $\alpha \in \mathbb{F}_q$ such that $H = g^\alpha$ and $h_{s,\gamma} = g^\alpha$. By Claim 2 and the fact that $\xi : [n] \to [n]$ is a bijection, there exists an $\ell \in [n]$ such that $\delta = \xi(\ell)$ and therefore $(\varphi(\ell), \xi(\ell), \cdot\,) \in R$. Since at round n one and only one perform a registration

$$(j, \xi(\ell), \cdot\,) \in R \quad \wedge \quad (\varphi(\ell), \xi(\ell), \cdot\,) \in R \quad \Rightarrow \quad j = \varphi(\ell).$$

Still by Claim 2 we deduce that $H = g^{x_{s,\ell}}$ so $\alpha = x_{s,\ell}$. However $h_{s,\gamma} = g_s^\alpha$ implies by Claim 1 that $\alpha = x_{s,\gamma}$. As all elements $h_{s,1}, \ldots, h_{s,n}$ are different by construction, we deduce that $\gamma = \ell$ and in particular, by definition of H_3 that $(eid, \varphi(\gamma)) = (eid, j) \in E$. Hence the aforementioned case only occurs with negligible probability.

Claim 4. $\mathsf{H}_4 \equiv \mathsf{H}_5$: We show that the behaviour of the functionalities is identical. Indeed if an honest P_j return $(\mathsf{claim}, eid, \delta, \pi)$ is H_4 then there is some $x \in \mathcal{K}$ such that

$$(j, \delta, H) \in R, \qquad h_{s,\gamma} \;=\; g_s^x, \qquad H \;=\; g^x.$$

Since ξ is a bijection there exists ℓ such that $\xi(\ell) = \delta$ and, as in previous claim, this implies $j = \varphi(\ell)$. Applying Claim 2 $H = g^{x_{s,\ell}}$ while from Claim 1 $h_{s,\gamma} = g^{x_{s,\gamma}}$ which implies $x = x_{s,\ell} = x_{s,\gamma} \;\Rightarrow\; \ell = \gamma$. In particular $(eid, \varphi(\gamma)) = (eid, j) \in E$.

Conversely if $(eid, j) \in E$ then $\varphi(\gamma) = j$ and by Claim 2, $(\varphi(\gamma), \xi(\gamma), H) \in R$ implies $H = g^{x_{s,\gamma}}$. Since by construction P_j stores in $\mathcal{K}$ the discrete logarithm of H in base g, $x_{s,\gamma} \in \mathcal{K}$ which, by Claim 1 also satisfies $h_{s,\gamma} = g_s^{s,\gamma}$. This concludes the claim's proof.

Claim 5. $\mathsf{DDH} \;\Rightarrow\; \mathsf{H}_6 \equiv \mathsf{H}_5$. Given $\mathcal{Z}$ a distinguisher which perform at most U elections, we will prove the claim through a sequence of intermediate functionalities H^*_σ which behave as H_6 until the σ-th shuffle (exclusive) and as H_5 from the σ-th shuffle on. For $\mathcal{Z}$ it is impossible to distinguish H_6 from H^*_U and by definition $\mathsf{H}^*_0 = \mathsf{H}_5$, thus it suffices to show $\mathsf{H}^*_\sigma \equiv \mathsf{H}^*_{\sigma+1}$. We reduce their indistinguishability to DDH_V through the algorithm $\mathcal{A}$ described below.

Description of $\mathcal{A}$: On input $(g, w_0, \ldots, w_V, \widetilde{g}, \widetilde{w}_0, \ldots, \widetilde{w}_V)$ tuple of group elements and $M \xleftarrow{\$} \mathcal{Z}$ the subset of corrupted parties $M \subseteq [N]$, initialise the following variables $\rho \leftarrow 1$; $s, n \leftarrow 0$, and functions $\xi : [n] \to [n]$, $\varphi : [n] \to [N]$. Next simulate H^*_σ with the following modifications

- When P_j registers, if it is a corrupted user, i.e. $j \notin M$, store $x_{s,n} \in \mathbb{F}_q$ such that $H = g^{x_{s,n}}$ and $h_{s,n} = g_s^{x_{s,n}}$. Else, if $j \in M$ sample $x_{s,n} \xleftarrow{\$} \mathbb{F}_q$ and set $H \leftarrow w_n^{x_{s,n}}$ and $h_{s,n} \leftarrow w_n^{\rho x_{s,n}}$.
- When a shuffle occurs let r, η be the witness either extracted from the proof of a corrupted user or sampled by an honest one. Update for all $\ell \in [n]$

$$\rho \leftarrow \rho \cdot r, \qquad \xi \leftarrow \xi \circ \eta, \qquad \varphi \leftarrow \varphi \circ \eta, \qquad x_{s+1,\ell} \leftarrow x_{s,\eta(\ell)}$$

Moreover, if the shuffler is honest, simulate the shuffle according to the cases below

$s < \sigma$: For all $\ell \in [n]$ set

$$h_{s+1,\ell} \leftarrow \begin{cases} h^r_{s,\eta(\ell)} & \text{If } \varphi(\ell) \in M \\ w^{\rho x_{s+1,n}}_{\xi(\ell)}, \quad x_{s+1,\ell} \leftarrow^{\$} \mathbb{F}_q & \text{If } \varphi(\ell) \notin M \end{cases}$$

$s = \sigma$: Fix $r = 1$, compute $g_{s+1} \leftarrow \widetilde{g}^\rho$ and set

$$h_{s+1,\ell} \leftarrow \begin{cases} g^{x_{s+1,n}}_{s+1} & \text{If } \varphi(\ell) \in M \\ \widetilde{w}^{\rho x_{s+1,n}}_{\xi(\ell)} & \text{If } \varphi(\ell) \notin M \end{cases}$$

$s > \sigma$: Simulate the shuffle as in H^*_σ

Finally when $b \leftarrow^{\$} \mathcal{Z}$, return b and halt.

Proof of Claim. First we observe that by induction one can prove that for all[7] $s < \sigma$ and $\ell \in [n]$ the group element $h_{s,\ell}$ has the form

$$\varphi(\ell) \in M \;\Rightarrow\; h_{s,\ell} = g_s^{x_{s,\ell}}, \qquad \varphi(\ell) \notin M \;\Rightarrow\; h_{s,\ell} = w^{\rho x_{s,\ell}}_{\xi(\ell)}.$$

Next we show that $\mathcal{A}$ perfectly simulates all those phases that are identical in H^*_σ and $\mathsf{H}^*_{\sigma+1}$. These are

– *Registration phase*: For honest users, we have that the discrete logarithm of H in base g and of $h_{s,n}$ in base g_s are equal since by construction $g_s = g^\rho$ and calling $\theta_n \in \mathbb{F}_q$ such that $g^{\theta_n} = w_n$ then $H = g^{\theta_n x_{s,n}}$ and $h_{s,n} = g_s^{\theta_n x_{s,n}}$

– *Shuffle phase, $s < \sigma$*: group elements linked to malicious users are correctly shuffled, while for honest users since $x_{s+1,n}$ is sampled randomly, $h_{s+1,n} \sim U(\mathbb{G})$ and independently from previous messages as specified in H_6

– *Shuffle phase, $s > \sigma$*: as in H^*_σ or H^*_σ by construction.

Finally, regarding the shuffle at round σ with an honest shuffler, we differentiate according to the kind of tuple $\mathcal{A}$ receives in input. In DDH^1_V there exists an $r \in \mathbb{F}_q$ such that $\widetilde{g} = g^r$ and $\widetilde{w}_i = w_i^r$. This implies that

$$\begin{aligned} \varphi(\ell) \in M \;\Rightarrow\; h_{s+1,\ell} \;&=\; g^{x_{s+1,\ell}}_{s+1} \;=\; g_s^{r x_{s,\eta(\ell)}} \;=\; h^r_{s,\eta(\ell)} \\ \varphi(\ell) \notin M \;\Rightarrow\; h_{s+1,\ell} \;&=\; \widetilde{w}^{\rho x_{s+1,\ell}}_{\xi_{s+1}(\ell)} \;=\; \widetilde{w}^{r\rho x_{s,\eta(\ell)}}_{\xi_s(\eta(\ell))} \;=\; h^r_{s,\eta(\ell)} \end{aligned}$$

where in the second case we let ξ_s be the value of ξ at round s. Hence $\mathcal{A}$ perfectly simulates H^*_σ. Conversely in DDH^0_V we have that group elements linked to malicious users are still correctly computed by calling $r \in \mathbb{F}_q$ the discrete logarithm of $\widetilde{g}$ in base g. For elements linked to honest users instead, since $\widetilde{w}_\ell$ are uniformly random and independent from w_ℓ and in particular from previous messages, $h_{s+1,\ell} \sim U(\mathbb{G})$. Thus $\mathcal{A}$ perfectly simulates $\mathsf{H}^*_{\sigma+1}$.

We can therefore conclude that $\mathsf{Adv}(\mathcal{A}) = \mathsf{Adv}(\mathcal{Z})$ and, as the first term is negligible by the DDH assumption, so is the second one.

[7] not only after an honest shuffle

Claim 6. $\mathsf{H}_6 \equiv \mathsf{H}_7$: Calling for each election j_0 and γ_0 the element sampled in H_6 and j_1, γ_1 the ones in H_7 we will prove they have the same distribution. Studying γ_b alone we have by construction that $\gamma_0 \sim U([n])$ while $\gamma_1 \sim U([n])$ because for all $\gamma^* \in [n]$

$$\begin{aligned}\Pr[\gamma_1 = \gamma^*] &= \sum_{j^* \in [N]} \Pr[\gamma_1 = \gamma^* | j = j^*] \cdot \Pr[j = j^*] \\ &= \Pr[\gamma_1 = \gamma^* | j = \varphi(\gamma^*)] \cdot \Pr[j = \varphi(\gamma^*)] \\ &= \frac{1}{|\varphi^{-1}(\varphi(\gamma^*))|} \cdot \frac{|\varphi^{-1}(\varphi(\gamma^*))|}{n} = \frac{1}{n}.\end{aligned}$$

Next, conditioning on $\gamma_b = \gamma^*$ for $b \in \{0,1\}$ we have that $j_0 = \varphi(\gamma^*)$ by construction while $\gamma_1 \in \varphi^{-1}(j_1)$ implies $\gamma^* \in \varphi^{-1}(j_1)$ that is $j_1 = \varphi(\gamma^*)$. By Proposition 3 we can conclude $\Delta((\gamma_0, j_0), (\gamma_1, j_1)) = 0$ as claimed.

Claim 7.. $\mathsf{H}_7 \equiv \mathsf{H}_8$: We will show that during each election the random variables (γ, δ) in both world follows the same distribution. A key observation in this direction is that after any honest shuffle, the permutation $\widehat{\eta}$ chosen is not fully revealed since for all $\ell \in [n]$ such that $\varphi(j) \notin M$, $h_{s+1,\ell}$ is uniformly random and does not depends on $\widehat{\eta}$. More precisely, calling $\widehat{\varphi} = \widetilde{\varphi} \circ \widehat{\eta}$ and $\widehat{\xi} = \widetilde{\xi} \circ \widehat{\eta}$, then the only informations provided to the adversary are the values of $\widehat{\eta}^{-1}$ on $\widetilde{\varphi}^{-1}(M)$. Therefore, defined K the set of all permutation that agrees with this information

$$K = \left\{ \eta \in S_n \ : \ \eta^{-1}_{|A} = \widehat{\eta}_{|A}, \ A = \widetilde{\varphi}^{-1}(M) \right\}$$

we have that, from the adversary's perspective, the random variable η representing the permutation chosen by the functionality is uniform over K. Setting $m = |[n] \setminus \widetilde{\varphi}^{-1}(M)|$ the number of group elements linked to honest users, is easy to find a bijection between K and S_m implying that $|K| = m!$. We further define two useful subsets of K: $K_{a,j}$ as the sets of permutations that send the index a to another element previously linked to P_j, and $K_{a,b}$ as the set of all function that sends a to b. Formally

$$K_a^j = \{\eta \in K \ : \ \eta(a) \in \widetilde{\varphi}^{-1}(j)\}, \qquad K_{a,b} = \{\eta \in K \ : \ \eta(a) = b\}.$$

Again is easy to show that if $a \notin \widehat{\varphi}^{-1}(M)$, $j \notin M$ and $b \notin \widetilde{\varphi}^{-1}(M)$ then the two sets have size respectively $|K_a^j| = |\widetilde{\varphi}^{-1}(j)| \cdot (m-1)!$ and $|K_{a,b}| = (m-1)!$. On the other hand, when $a \in \widehat{\varphi}^M$ and $j \notin M$ we have that $K_a^j = \varnothing$ since if by contradiction a permutation η belongs to this set

$$\widehat{\eta}(a) \in \widetilde{\varphi}^{-1}(M) \quad \Rightarrow \quad a = \widehat{\eta}^{-1}(\widehat{\eta}(a)) = \eta^{-1}(\widehat{\eta}(a)) \quad \Rightarrow \quad \eta(a) = \widehat{\eta}(a),$$

which is a contradiction since the first term belongs to $\widetilde{\varphi}^{-1}(j)$ while the second one to $\widetilde{\varphi}^{-1}(M)$ in spite of the two sets being disjoint. Finally we will denote $\varphi = \eta \circ \widetilde{\varphi}$ and $\xi = \eta \circ \widetilde{\xi}$.

Next we study the statistical distance of (γ_0, δ_0) and (γ_1, δ_1) under the simplifying assumption that an honest shuffle occurred in the last step, where the two couples represents the values of γ, δ in H_7 and H_8 respectively. The first trivial case is when the sampled j such that $(eid, j) \in E$ lies in M. In this case both γ_0 and γ_1 follow the same probability distribution by construction and δ_0, δ_1 are returned by the adversary.

Conversely if $j \notin M$, for any $\gamma^* \in [n]$ we observe that $\gamma_0 = \gamma^*$ implies $\gamma^* \in \varphi^{-1}(j) = \eta^{-1}(\widetilde{\varphi}^{-1}(j))$. In particular $\eta(\gamma^*) \in \widetilde{\varphi}^{-1}(j)$ which means that a necessary condition for $\gamma_0 = \gamma^*$

is $\eta \in K^j_{\gamma^*}$ from which we derive $\Pr[\gamma_0 = \gamma^*] = 0$ when $\gamma^* \in \widehat{\varphi}^{-1}(M)$ as in this case $K^j_{\gamma^*} = \varnothing$ as proved previously. Instead, when $\gamma^* \notin \widehat{\varphi}^{-1}(M)$

$$\begin{aligned}
\Pr[\gamma_0 = \gamma^*] &= \sum_{\eta^* \in K} \Pr[\gamma_0 = \gamma^* | \eta = \eta^*] \Pr[\eta = \eta^*] \\
&= \sum_{\eta^* \in K^j_{\gamma^*}} \Pr[\gamma_0 = \gamma^* | \eta = \eta^*] \Pr[\eta = \eta^*] \\
&= \sum_{\eta^* \in K^j_{\gamma^*}} \frac{1}{|(\widetilde{\varphi} \circ \eta^*)^{-1}(j)|} \cdot \frac{1}{m!} \\
&= \sum_{\eta^* \in K^j_{\gamma^*}} \frac{1}{|\widetilde{\varphi}^{-1}(j)|} \cdot \frac{1}{m!} = \frac{|\widetilde{\varphi}^{-1}(j)| \cdot (m-1)!}{|\widetilde{\varphi}^{-1}(j)| \cdot m!} = \frac{1}{m}.
\end{aligned}$$

Regarding γ_1 we have that $\gamma_1 \sim U([n] \setminus \widehat{\varphi}^{-1}(M))$ where we remark that by the definition of K, $\widehat{\varphi}^{-1}(M) = \varphi^{-1}(M)$ i.e. the preimage of M does not depends on the choice of η. In particular when $\gamma^* \in \widehat{\varphi}^{-1}(M)$ the probability that $\gamma_1 = \gamma^*$ is zero, while otherwise is $1/m$ by construction. It follows that γ_0, γ_1 have the same distribution.

Using Proposition 3 we can move on to study δ_0, δ_1 conditioned on $\gamma_0 = \gamma^*$ and $\gamma_1 = \gamma^*$. We notice en passant that the first condition restricts the conditional variable $\eta \sim U(K^j_{\gamma^*})$. For any $\delta^* \in [n]$ the condition $\delta^* = \delta_0$ implies that $\delta^* = \widetilde{\xi}(\eta(\gamma^*))$ that is $\eta(\gamma^*) = \widetilde{\xi}^{-1}(\delta^*)$. This means that η must map γ^* to $\widetilde{\xi}^{-1}(\delta^*)$ even though the condition $\eta \in K^j_{\gamma^*}$ restrict the values $\eta(\gamma^*)$ can have to the set $\widetilde{\varphi}^{-1}(j)$. Hence, calling $\beta = \widetilde{\xi}^{-1}(\delta^*)$ for notational purposes, a necessary condition for $\delta_0 = \delta^*$ is $\beta \in \widetilde{\varphi}^{-1}(j)$ and $\eta \in K_{\gamma^*,\beta}$

$$\begin{aligned}
\Pr[\delta_0 = \delta^*] &= \sum_{\eta \in K^j_{\gamma^*}} \Pr[\delta_0 = \delta^* | \eta = \eta^*] \Pr[\eta = \eta^*] \\
&= \sum_{\eta \in K_{\gamma^*,\beta}} \Pr[\delta_0 = \delta^* | \eta = \eta^*] \Pr[\eta = \eta^*] \\
&= \sum_{\eta \in K_{\gamma^*,\beta}} \frac{1}{|\widetilde{\varphi}^{-1}(j)| \cdot (m-1)!} = \frac{(m-1)!}{|\widetilde{\varphi}^{-1}(j)| \cdot (m-1)!} = \frac{1}{|\widetilde{\varphi}^{-1}(j)|}.
\end{aligned}$$

By construction instead $\delta_1 \xleftarrow{\$} \xi(\varphi^{-1}(j)) = \widetilde{\xi}(\widetilde{\varphi}(j))$ hence again in order to have $\delta_1 = \delta^*$ a necessary condition is $\widetilde{\xi}^{-1}(\delta^*) \in \widetilde{\varphi}^{-1}(j)$. Moreover when this condition is met, since $\widetilde{\xi}$ is a bijection we have that $\widetilde{\xi}^{-1}(\delta) \sim U(\widetilde{\varphi}^{-1}(j))$, hence

$$\Pr[\delta_1 = \delta^*] = \Pr\left[\widetilde{\xi}(\delta_1) = \widetilde{\xi}(\delta^*)\right] = \frac{1}{|\widetilde{\varphi}^{-1}(j)|}.$$

The claim is therefore proven for the specific case when the election occurs after an honest shuffle. For the general case instead, assume that the last honest shuffle occurred at step σ. Then the proof follows, observing that in intermediate elections no honest party wins i.e. no information on η_σ is revealed, with the same argument up to applying $\eta_{\sigma+1} \circ \ldots \circ \eta_s$ to K, K^j_a and $K_{a,b}$ to the right.

B.3 Adaptive Construction

Notation: Throughout the proof we denote $R_j = \{\delta : (j, \delta) \in R\}$, where R is the set that keeps track of registrations in Protocol 4

Proof (of Theorem 2). As in the static case, the main challenge in the construction of a simulator is to correctly reproduce the election phase. To this aim we exploit the UC ZK-proof provided at registration time to let $\mathcal{S}$ extract (α, δ) such that $h = \mathbf{g}^{(\alpha,\delta)}$. In this way it is possible to link commitments in the list at any given round to malicious users, allowing $\mathcal{S}$ to correctly reproduce elections in which a corrupted user wins.

Conversely, commitments linked to users not yet corrupted are independently randomized each time an honest player performs a shuffle. In this way if $\mathcal{F}_{\mathsf{SSLE}}$ says at election time that the winner is honest, $\mathcal{F}_{\mathsf{SSLE}}$ can choose a random commitment in the list linked to uncorrupted users. Finally, corruption is carried out by equivocating the commitment.

Simulator $\mathcal{S}$: Initialise $M \leftarrow \varnothing$ the set of corrupted parties, $s \leftarrow 0$ round counter, $R \leftarrow \varnothing$ list of registered users, $n \leftarrow 0$ size of R and $\xi : [n] \to [n]$ the composition of all the permutations applied when shuffling. Sample $\mathbf{y} \xleftarrow{\$} \mathbb{F}_q^2$, trapdoor of the Pedersen commitment and set $\mathbf{g}_0 \leftarrow g^{\mathbf{y}}$. When parties initially query $\mathcal{F}_{\mathsf{ct}}$ return $\mathbf{g}_0$. Upon receiving:

– $(\mathsf{registered}, j)$ from $\mathcal{F}_{\mathsf{SSLE}}$, with $j \notin M$:
  - Sample $\mathbf{z}_{s,n} \xleftarrow{\$} \mathbb{F}_q^2$ and set $h_{s,n} \leftarrow \mathbf{g}_s^{\mathbf{z}_{s,n}}$
  - Broadcast $(\mathsf{proof}, n, \mathbf{g}_s, h_{s,n}, n)$ as $\mathcal{F}_{\mathsf{zk}}^{\mathcal{R}_{\mathsf{ped}}}$
  - Execute the shuffle procedure (see below)

– $(\mathsf{prove}, sid, n, \mathbf{g}_s, h_{s,n}, \alpha)$ from corrupted P_j:
  - If $\mathbf{g}_s^{(\alpha,n)} = h_{s,n}$, store $\mathbf{z}_{s,n} \leftarrow (\alpha, n)$
  - Broadcast $(\mathsf{proof}, sid, n, \mathbf{g}_s, h_{s,n})$
  - Update $R \leftarrow R \cup \{(j, n)\}$, $n \leftarrow |R|$
  - Send $(\mathsf{register})$ to $\mathcal{F}_{\mathsf{SSLE}}$ as P_j

– $(\mathsf{shuffle}, sid, \mathbf{g}_{s+1}, \mathbf{h}_{s+1}, \mathbf{k}_{s+1}, \pi)$ from corrupted P_j:
  - Check the correctness of π
  - Determine $\eta \in S_n$ such that $h_{s+1,\ell} = \mathbf{g}_{s+1}^{\mathbf{z}_{s,\eta(\ell)}}$ for all $\ell \in [n]$
  - Update $\xi \leftarrow \xi \circ \eta$ and $\mathbf{z}_{s+1,\ell} \leftarrow \mathbf{z}_{s,\eta(\ell)}$

– $(\mathsf{update}, sid, u_{s+1,\delta})$ from P_j:
  If π is accepted and $(j, \delta) \in R$, set $k_{s+1,\delta} \leftarrow k_{s,\delta} \cdot u_{s+1,\delta}$.

– A request from $\mathcal{F}_{\mathsf{SSLE}}$ to send $(\mathsf{outcome}, eid)$:
  - If a corrupted user would receive $(\mathsf{outcome}, eid, 1)$: sample $\delta \xleftarrow{\$} R_j$ and set $\gamma \leftarrow \xi^{-1}(\delta)$.
  - Else, sample $(j, \delta) \xleftarrow{\$} R$ such that P_j is not corrupted, i.e. $j \notin M$ and set $\gamma \leftarrow \xi^{-1}(\delta)$.
  - Broadcast $(\mathsf{tossed}, eid, \gamma)$ as $\mathcal{F}_{\mathsf{ct}}^n$ and allow $\mathcal{F}_{\mathsf{SSLE}}$ to send the requested messages.

– $(\mathsf{result}, eid, j)$ from $\mathcal{F}_{\mathsf{SSLE}}$, with P_j honest:

  - Sample $\delta \xleftarrow{\$} R_j$, simulate π and broadcast $(\mathsf{claim}, eid, \delta, \pi)$ as P_j
  - Call $\gamma' \leftarrow \xi^{-1}(\delta)$, $\delta' \leftarrow \xi(\gamma)$. Swap $\xi(\gamma) \leftarrow \delta$ and $\xi(\gamma') \leftarrow \delta'$
  - Execute the shuffle procedure (see below)

– $(\mathsf{rejected}, eid, j)$ from $\mathcal{F}_{\mathsf{SSLE}}$, with P_j honest: Broadcast $(\mathsf{claim}, eid, \perp)$ as P_j.

– $(\mathsf{claim}, eid, \delta, \pi)$ from corrupted P_j:
  - If π is accepted and $(\delta, j) \in R$, send (reveal, eid) as P_j to $\mathcal{F}_{\mathsf{SSLE}}$
  - Else, send $(\mathsf{fake\_rejected}, eid, j)$ to $\mathcal{F}_{\mathsf{SSLE}}$

– $(\mathsf{corrupt}, j)$ from $\mathcal{Z}$:
  - Add j to the set of corrupted users $M \leftarrow M \cup \{j\}$
  - Send $(\mathsf{corrupt}, j)$ to $\mathcal{F}_{\mathsf{SSLE}}$ and wait for $(\mathsf{corrupted}, j, E_j)$
  - If no one claimed election eid, $\mathcal{F}^n_{\mathsf{ct}}$ sent $(\mathsf{tossed}, eid, \gamma)$, and $eid \in E_j$: get $\delta \xleftarrow{\$} R_j$, call $\gamma' \leftarrow \xi^{-1}(\delta)$, $\delta' \leftarrow \xi(\gamma)$ and swap $\xi(\gamma) \leftarrow \delta$, $\xi(\gamma') \leftarrow \delta'$
  - Initialise $\mathcal{K}_j \leftarrow \varnothing$ the set of simulated keys belonging to P_j
  - For every $\delta \in R_j$, calling $\gamma \leftarrow \xi^{-1}(\delta)$ find α such that

$$\mathbf{y}^\top(\alpha, \delta) \; = \; \mathbf{y}^\top(\mathbf{z}_{s,\gamma} + \mathbf{w}_{s,\delta})$$

    and add $\mathcal{K}_j \leftarrow \mathcal{K}_j \cup \{(\alpha, \delta)\}$
  - Send $(\mathsf{corrupted}, j, \mathcal{K}_j)$ to the environment $\mathcal{Z}$

Shuffle sub-procedure with session ID sid:

1. Sample a permutation $\eta \xleftarrow{\$} S_n$ and $r \xleftarrow{\$} \mathbb{F}_q$.
2. Update $\xi \leftarrow \xi \circ \eta$ and set $\mathbf{g}_{s+1} \leftarrow \mathbf{g}_s^r$.
3. For all $\ell \in [n]$, calling $\delta = \xi(\ell)$:
   If the ℓ-th commitment is linked to an honest user, i.e. if there exists $j \notin M$ such that $(j, \delta) \in R$, find $\alpha \in \mathbb{F}_q$ such that $\mathbf{y}^\top(\alpha, \delta) = \mathbf{y}^\top(\mathbf{z}_{s,\eta(\ell)} + \mathbf{w}_{s,\delta})$ and set

$$\mathbf{z}_{s+1,\ell} \xleftarrow{\$} \mathbb{F}_q^2, \qquad h_{s+1,\ell} \leftarrow \mathbf{g}_{s+1}^{\mathbf{z}_{s+1,\ell}},$$
$$\mathbf{w}_{s+1,\delta} \leftarrow (\alpha, \delta) - \mathbf{z}_{s+1,\ell}, \qquad k_{s+1,\delta} \leftarrow \mathbf{g}_{s+1}^{\mathbf{w}_{s+1,\delta}}.$$

   Otherwise, if the ℓ-th commitment is linked to a corrupted user

$$\mathbf{z}_{s+1,\ell} \leftarrow \mathbf{z}_{s,\eta(\ell)}, \quad h_{s+1,\ell} \leftarrow h^r_{s,\eta(\ell)}, \quad k_{s+1,\delta} \leftarrow k^r_{s,\delta}.$$

4. Simulate π
5. Broadcast $(\mathsf{shuffle}, sid, \mathbf{g}_{s+1}, \mathbf{h}_{s+1}, \mathbf{k}_{s+1}, \pi)$
6. After the adversary deliveries the previous message, for all indices $\delta \in [n]$ associated to an honest user[8]: $\mathbf{w}_{s+1,\delta} \xleftarrow{\$} \mathbb{F}_q^2$, $u_{s+1,\delta} \leftarrow k^{-1}_{s+1,\delta} \cdot \mathbf{g}_{s+1}^{\mathbf{w}_{s+1,\delta}}$, simulate π and broadcast $(\mathsf{update}, sid, u_{s+1,\delta}, \pi)$ as P_j

[8] again, such that there exists $j \notin M \ : \ (j, \delta) \in R$

Next we define a sequence of hybrid functionalities to prove that the real protocol is indistinguishable from $\mathcal{F}_{\mathsf{SSLE}} \circ \mathcal{S}$ for any PPT environment $\mathcal{Z}$.

H_0: The real protocol

H_1: As the previous one, but all NIZK proof are simulated

H_2: As H_1 but when any user send at registration time $(\mathsf{prove}, sid, n, \mathbf{g}_s, h_{s,n}, n, \alpha)$ to $\mathcal{F}_{\mathsf{zk}}^{\mathcal{R}_{\mathsf{ped}}}$ correctly, store $\mathbf{z}_{s,n} \leftarrow (\alpha, n)$. Moreover, set initially $\xi : [n] \to [n]$. During any shuffle, if the shuffler is honest let η be the permutation chosen, otherwise find $\eta \in S_n$ such that $h_{s+1,\ell} = \mathbf{g}_{s+1}^{\mathbf{z}_{s,\eta(\ell)}}$. Set

$$\xi \leftarrow \xi \circ \eta, \quad \mathbf{z}_{s+1,\ell} \leftarrow \mathbf{z}_{s+1,\eta(\ell)}.$$

H_3: As H_2 when the environment corrupts P_j, set $\mathcal{K}^*_{s,j} \leftarrow \varnothing$. For all $\delta \in R_j$, calling $\gamma = \xi^{-1}(\delta)$, find $\alpha \in \mathbb{F}_q$ such that $\mathbf{g}_s^{(\alpha,\delta)} = h_{s,\gamma} \cdot k_{s,\delta}$ and add $\mathcal{K}^*_{s,j} \leftarrow \mathcal{K}^*_{s,j} \cup \{(\alpha, \delta)\}$. Return $(\mathsf{corrupted}, j, \mathcal{K}^*_{s,j})$ to the environment.

H_4: As H_3 but initially set $E \leftarrow \varnothing$. During the election eid, if the random beacon $\mathcal{F}_{\mathsf{ct}}^n$ return $(\mathsf{tossed}, eid, \gamma)$, set $\delta \leftarrow \xi(\gamma)$ and find j such that $(j, \delta) \in R$. Add $E \leftarrow E \cup \{(eid, j)\}$.

H_5: As H_4 but when a corrupted user P_j sends $(\mathsf{claim}, eid, \delta, \pi)$, honest users return $(\mathsf{result}, eid, j)$ if $(j, \delta) \in R$, π is accepted and $(eid, j) \in E$. Else they return $(\mathsf{rejected}, eid, j)$.

H_6: As H_5 but each honest user P_j, upon receiving (reveal, eid), check if $(eid, j) \in E$. If this is the case it broadcast $(\mathsf{claim}, eid, \delta, \pi)$ where π is simulated and $\delta = \xi(\gamma)$ with γ being the index returned by $\mathcal{F}_{\mathsf{ct}}^n$ for the election eid. Else, broadcast $(\mathsf{claim}, eid, \perp)$.

H_7: As H_6 but every time an honest user P_i performs a shuffle, for all $\ell \in [n]$ linked to honest users (i.e. such that calling $\delta = \xi(\ell)$ there is an honest user P_j s.t. $(j, \delta) \in R$) find $\alpha \in \mathbb{F}_q$ such that $h_{s,\eta(\ell)} \cdot k_{s,\delta} = \mathbf{g}_s^{(\alpha,\delta)}$ and set

$$\mathbf{z}_{s+1,\ell} \xleftarrow{\$} \mathbb{F}_q^2, \qquad h_{s+1,\ell} \leftarrow \mathbf{g}_{s+1}^{\mathbf{z}_{s+1,\ell}},$$
$$\mathbf{w}_{s+1,\delta} \leftarrow (\alpha, \delta) - \mathbf{z}_{s+1,\ell}, \qquad k_{s+1,\delta} \leftarrow \mathbf{g}_{s+1}^{\mathbf{w}_{s+1,\delta}}.$$

Furthermore, in the update phase honest users samples $u_{s+1,\delta} \xleftarrow{\$} \mathbb{G}$.

H_8: As H_7 but for each election sample $(j, \cdot\,) \xleftarrow{\$} R$ and add (eid, j) to E. Next, sample $\delta \xleftarrow{\$} R_j$, call $\gamma \leftarrow \xi^{-1}(\delta)$ and return $(\mathsf{tossed}, eid, \gamma)$.

H_9: As H_7 but when an honest party P_j with $(eid, j) \in E$ receives (reveal, eid) or is corrupted before receiving the reveal command:
Sample $\delta \xleftarrow{\$} R_j$, call $\gamma' \leftarrow \xi^{-1}(\delta)$ and $\delta' \leftarrow \xi(\gamma)$ and swap $\xi(\gamma) \leftarrow \delta$, $\xi(\gamma') \leftarrow \delta'$.

H_{10}: The simulated protocol $\mathcal{F}_{\mathsf{SSLE}} \circ \mathcal{S}$.

We immediately deal with trivial cases pointing out that $\mathsf{H}_0 \equiv \mathsf{H}_1$ by perfect zero-knowledge of the argument used while $\mathsf{H}_1 \equiv \mathsf{H}_2$ and $\mathsf{H}_3 \equiv \mathsf{H}_4$ are equivalent because they don't affect the behaviour of the functionality. Next, given a PPT environment $\mathcal{Z}$, which performs at most $U(\lambda)$ elections and $V(\lambda)$ registrations, we proceed to prove the following Lemmas.

Claim 1. *If the* DLP *is hard in* $\mathbb{G}$*, up to negligible probability at any step* s *in* H_0 *the commitments* $h_{s,1},\dots,h_{s,n}$ *are all distinct.* Given a PPT environment $\mathcal{Z}$ we build $\mathcal{A}$ breaking the DLP in $\mathbb{G}$.

Description $\mathcal{A}$: Initially it receives $(g,v)\in\mathbb{G}_2$. Samples $\theta\xleftarrow{\$}\mathbb{F}_q$ and sets $\mathbf{g}_0\leftarrow(g^\theta,v)$ the initial value returned by $\mathcal{F}_{\mathsf{ct}}$. When a party sends at registration time $(\mathsf{prove}, sid, n, \mathbf{g}_s, h_{s,n}, \alpha)$ with $h_{s,n}=\mathbf{g}_s^{(\alpha,n)}$, store $\mathbf{z}_{s,n}\leftarrow(\alpha,n)$. After a shuffle, extract from the NIZK proof a witness (r_s,η_s) and set $\mathbf{z}_{s+1,\ell}\leftarrow\mathbf{z}_{s+1,\eta_s(\ell)}$.
If at any point $h_{s,i}=h_{s,j}$ let $\mathbf{z}_{s,i}-\mathbf{z}_{s,j}=(\zeta_1,\zeta_2)$ and return $x\leftarrow-\theta\zeta_1\zeta_2^{-1}$.

Proof of Claim. By weak simulation extractability we have that after any shuffle, if the proof is accepted then (r_s,η_s) is a valid witness, meaning that $h_{s+1,\ell}=h_{s,\eta_s(\ell)}^{r_s}$. Hence, it is easy to prove by induction that until $\mathcal{A}$ does not halt $h_{s,\ell}=\mathbf{g}_0^{\rho_s\mathbf{z}_{s,\ell}}$ with $\rho_s=r_1\cdot\ldots\cdot r_s$. If at some point $h_{s,i}=h_{s,j}$ then

$$\mathbf{g}_0^{\rho_s(\mathbf{z}_{s,i}-\mathbf{z}_{s,j})}=1\quad\Rightarrow\quad\mathbf{g}_0^{\mathbf{z}_{s,i}-\mathbf{z}_{s,j}}=1\quad\Rightarrow\quad g^{\theta\zeta_1}\cdot v^{\zeta_2}=1\quad\Rightarrow\quad v=g^{-\theta\zeta_1\zeta_2^{-1}}$$

where ζ_2 is non zero because otherwise ζ_1 would be zero and in particular $\mathbf{z}_{s,i}=\mathbf{z}_{s,j}$ which is a contradiction since $\mathbf{z}_{s,i}$ is a non zero multiple of $(1,\xi(i))$ and $\mathbf{z}_{s,j}$ is a non zero multiple of $(1,\xi(j))$. We therefore conclude that $\mathcal{A}$ breaks DL if and only if a collision in the commitments occur.

Claim 2. *In* H_2 *for all* s *up to negligible probability, there exists a unique* $\eta_s\in S_n$ *such that* $h_{s+1,\ell}=\mathbf{g}_{s+1}^{\mathbf{z}_{s,\eta_s(\ell)}}$. We prove the statement, which is vacuously true for $s=0$, by induction. Assuming it true for all round before the $(s+1)$-th, we have that $h_{s,\ell}=\mathbf{g}_s^{\mathbf{z}_{s,\ell}}$. Calling $(\widetilde{r}_s,\widetilde{\eta}_s)$ the witness extracted by the proof of correct shuffling then

$$\mathbf{g}_{s+1}=\mathbf{g}_s^{\widetilde{r}_s}\quad\Rightarrow\quad h_{s+1,\ell}=h_{s,\widetilde{\eta}(\ell)}^{\widetilde{r}_s}=\mathbf{g}_s^{\widetilde{r}_s\mathbf{z}_{s,\widetilde{\eta}(\ell)}}=\mathbf{g}_{s+1}^{\mathbf{z}_{s,\widetilde{\eta}(\ell)}}$$

Next, due to the previous claim, $\mathbf{g}_{s+1}^{\mathbf{z}_{s+1,\widetilde{\eta}(\ell)}}$ is only equal to $h_{s+1,\ell}$. As a consequence $\widetilde{\eta}_s$ is the only permutation that satisfies the thesis.

Claim 3. *Let* $\mathcal{K}_{s,j}$ *be the keyring of an uncorrupted user* P_j *at round* s *in* H_2*. Then for all* $(\alpha,\delta)\in\mathcal{K}_{s,j}$*, calling* $\gamma=\xi^{-1}(\delta)$*,* $h_{s,\gamma}k_{s,\delta}=\mathbf{g}_s^{(\alpha,\delta)}$.

As before we prove the statement by induction. Let σ be the round in which P_j perform the δ-th registration, then $h_{\sigma,\delta}=\mathbf{g}_s^{(\alpha,\delta)}$ and $k_{\sigma,\delta}=1$. Assuming the thesis for $s\geq\sigma$ we show it holds for $s+1$. By the inductive hypothesis, calling $\xi_s^{-1}(\delta)=\gamma'$ we have that

$$(\alpha,\delta)\in\mathcal{K}_{s,j}\quad\Rightarrow\quad\mathbf{g}_s^{(\alpha,\delta)}=h_{s,\gamma'}\cdot k_{s,\delta}.$$

After the next shuffle, let η_{s+1} be the next permutation and r_{s+1} the randomness used, then calling $\gamma=\eta_{s+1}(\gamma')$

$$\mathbf{g}_{s+1}=\mathbf{g}_s^{r_{s+1}},\qquad h_{s+1,\gamma}=h_{s,\gamma'}^{r_{s+1}},\qquad k_{s+1,\delta}=k_{s,\delta}^{r_{s+1}}$$

Therefore α opens the Pedersen commitment $h_{s+1,\gamma}k_{s+1,\delta}$ to δ since

$$\mathbf{g}_{s+1}^{(\alpha,\delta)}=\mathbf{g}_s^{r_{s+1}(\alpha,\delta)}=h_{s,\gamma'}^{r_{s+1}}\cdot k_{s,\delta}^{r_{s+1}}=h_{s+1,\gamma}\cdot k_{s+1,\delta}.$$

In order to conclude the proof we need to show that the relations holds even after the update. Let ω be the exponent P_j uses to generate $u_{s+1,\delta} = \mathbf{g}_{s+1}^{(\omega,0)}$. Then after the update $\alpha' = \alpha + \omega$ with $(\alpha', \delta) \in \mathcal{K}_{s+1,j}$ and $k'_{s+1,\delta} = k_{s+1,\delta} \cdot u_{s+1,\delta}$ implies that

$$\mathbf{g}_{s+1}^{(\alpha',\delta)} \;=\; \mathbf{g}_{s+1}^{(\alpha,\delta)} \cdot \mathbf{g}_{s+1}^{(\omega,0)} \;=\; h_{s+1,\gamma} \cdot k_{s+1,\delta} \cdot u_{s+1,\delta} \;=\; h_{s+1,\gamma} \cdot k'_{s+1,\delta}.$$

Claim 4. $\mathsf{H}_2 \equiv \mathsf{H}_3$. The only difference between them occurs when a honest party P_j is corrupted. Let $\mathcal{K}_{s,j}$ be the set of keys stored by P_j at round s and $\mathcal{K}^*_{s,j}$ the set of keys returned by H_3 when P_j is corrupted at round s. We will conclude by proving $\mathcal{K}_{s,j} = \mathcal{K}^*_{s,j}$.

If $(\alpha, \delta) \in \mathcal{K}_{s,j}$ by the previous claim, calling $\gamma = \xi^{-1}(\delta)$, $h_{s,\gamma} \cdot k_{s,\delta} = \mathbf{g}_s^{(\alpha,\delta)}$ therefore $(\alpha, \delta) \in \mathcal{K}^*_{s,j}$.

Vice versa if $(\alpha, \delta) \in \mathcal{K}^*_{s,j}$, then calling $\gamma = \xi^{-1}(\delta)$, we have by construction that $h_{s,\gamma} \cdot k_{s,\delta} = \mathbf{g}_s^{(\alpha,\delta)}$ and $(j, \delta) \in R$. The second fact means that in some step P_j performed the δ-th registration, hence there exists a $(\beta, \delta) \in \mathcal{K}_{s,j}$ and by the previous claim $h_{s,\gamma} \cdot k_{s,\delta} = \mathbf{g}_s^{(\beta,\delta)}$. Letting $r_1, \ldots, r_s$ be the exponents used in the shuffle rounds, and $\rho_s = r_1 \cdot \ldots \cdot r_s$

$$\mathbf{g}_s^{(\alpha,\delta)} \;=\; \mathbf{g}_s^{(\beta,\delta)} \quad\Rightarrow\quad 1 \;=\; \mathbf{g}_s^{(\alpha-\beta,0)} \;=\; \mathbf{g}_0^{\rho_s(\alpha-\beta,0)}$$

Up to negligible probability the first component of $\mathbf{g}_0$ is non zero and $\rho_s \neq 0$, so we can conclude $\alpha = \beta$ and in particular $(\alpha, \delta) \in \mathcal{K}_{s,j}$. The Claim is thus proven.

Claim 5. $\mathsf{DLP} \Rightarrow \mathsf{H}_5 \equiv \mathsf{H}_4$: The only difference is that in H_5, when a dishonest user broadcast a claim message, honest ones further check that $(eid, j) \in E$. Hence an adversary can distinguish the two world if he manage to generate $(\mathsf{claim}, eid, \delta, \pi)$ such that $(j, \delta) \in R$, π is accepted by the NIZK verifier but $(eid, j) \notin E$. Calling bad this event, we show that whenever bad occurs, the environment's knowledge can be used to break the DLP over $\mathbb{G}$. Formally we perform this reduction through an algorithm $\mathcal{B}$.

Description of $\mathcal{B}$. Initially it receives (g, v) instance of the DLP. Sample $\theta \leftarrow^{\$} \mathbb{F}_q$ and set $\mathbf{g}_0 \leftarrow (g^\theta, v)$ the initial vector returned by $\mathcal{F}_{\mathsf{ct}}$. Next, it simulates H_4 with the following changes:

- At corruption time it returns the set of keys belonging to P_j in $\mathcal{K}_j$ instead of equivocating them
- When an adversary performs a shuffle extract (r_s, η_s) form the NIZK proof, otherwise let (r_s, η_s) be the field element and permutation used by the honest shuffler.
- When an adversary perform a correct key update $(\mathsf{update}, sid, u_{s,\delta}, \pi)$ extract ω from π such that $u_{s,\delta} = \mathbf{g}_s^{(\omega,0)}$ and set $\mathbf{v}_{s+1,\delta} \leftarrow \mathbf{v}_{s,\delta} + (\omega, 0)$
- When a corrupted user P_j sends $(\mathsf{claim}, eid, \delta, \pi)$ such that bad occurs: Extract α from π and set $(\alpha, \delta) - \mathbf{z}_{s,\gamma} - \mathbf{v}_{s,\delta} = (\zeta_1, \zeta_2)$ where γ is the index returned by $\mathcal{F}^n_{\mathsf{ct}}$ for the election eid. Return $x \leftarrow -\theta\zeta_1\zeta_2^{-1}$.

When $\mathcal{Z}$ halts, return $\perp$.

Proof of Claim. First we remark that $\mathcal{B}$ perfectly simulates H_4 and H_5, which until bad occurs are identical. The only point to clarify here is that during player corruption it does not simulates the keys but instead it returns the real one - which however up to negligible probability produce the same result as shown in Claim 2. Next we remark that by induction one can easily show $k_{s,\delta} = \mathbf{g}_s^{\mathbf{v}_{s,\delta}}$, which follow since in the protocol $k_{s+1,\delta} = k_{s,\delta} \cdot u_{s,\delta}$.

If bad occurs at election eid, let $(\mathsf{tossed}, eid, \gamma)$ be the message returned by $\mathcal{F}_{\mathsf{ct}}^n$, $\delta' = \xi(\gamma)$ and $j' \in [N]$ such that $(j', \delta') \in R$. By construction then $(eid, j') \in E$. By the way we defined bad, $(eid, j) \notin E$, so $j \neq j'$. This in turns implies that $\delta \neq \delta'$ or by contradiction we would have $(j', \delta), (j, \delta) \in R$ which would yield $j' = j$.

Calling σ the round in which $P_{j'}$ performed the δ'-th registration, the exponent used to generate $h_{\sigma,\delta'}$ is of the form $\mathbf{z}_{\sigma,\delta} = (\beta, \beta\delta')$. By previous claims

$$\delta' = \xi^{-1}(\gamma) \quad \Rightarrow \quad h_{s,\gamma} \; = \; \mathbf{g}_s^{\mathbf{z}_{s,\gamma}} \; = \; \mathbf{g}_s^{\mathbf{z}_{\sigma,\delta'}} \; = \; \mathbf{g}_s^{(\beta,\delta')}$$

However, by the simulation extractability of the proof in the final claim message sent by P_j we also have that

$$\mathbf{g}_s^{(\alpha,\delta)} \; = \; h_{s,\gamma} \cdot k_{s,\delta} \; = \; \mathbf{g}_s^{(\beta,\delta')} \cdot \mathbf{g}_s^{\mathbf{v}_{s,\delta}}$$

It can be easily shown that $\mathbf{v}_\delta = (\omega, 0)$ for some ω in $\mathbb{F}_q$, which implies $\zeta_1 = \alpha - \beta - \omega$, $\zeta_2 = \delta - \delta' \neq 0$. Moreover, from the previous equation $1 = \mathbf{g}_s^{(\zeta_1,\zeta_2)}$ implying

$$\mathbf{g}_0^{(\zeta_1,\zeta_2)} \; = \; 1 \quad \Rightarrow \quad g^{\theta\zeta_1} v^{\zeta_2} \; = \; 1 \quad \Rightarrow \quad v = g^{-\theta\zeta_1\zeta_2}.$$

This concludes the reduction.

Claim 6. $\mathsf{H}_6 \equiv \mathsf{H}_5$. We will prove that the two world are identical. To this aim it is enough to show that $(eid, j) \in E$ with P_j currently honest, if and only if P_j, meaning that there exists $(\alpha, \delta) \in \mathcal{K}_{s,j}$ such that $h_{s,\gamma} \cdot k_{s,\delta} = \mathbf{g}_s^{(\alpha,\delta)}$, where $(\mathsf{tossed}, eid, \gamma)$ is the message returned by $\mathcal{F}_{\mathsf{ct}}^n$.

If $(eid, j) \in E$, then $\mathcal{F}_{\mathsf{ct}}^n$ returned $(\mathsf{tossed}, eid, \gamma)$ with $\xi(\gamma) = \delta$ and $(j, \delta) \in R$ which means that at round $\sigma \leq s$, P_j performed the δ-th registration. Hence $(\alpha', \delta) \in \mathcal{K}_{\sigma,j}$ and by induction $(\alpha, \delta) \in \mathcal{K}_{s,j}$ for some $\alpha \in \mathbb{F}_q$. By previous claims then $\mathbf{g}_s^{(\alpha,\delta)} = h_{s,\gamma} \cdot k_{s,\delta}$.

Conversely assume there exists $(\alpha, \delta) \in \mathcal{K}_{s,j}$ such that $h_{s,\gamma} \cdot k_{s,\delta} = \mathbf{g}_s^{(\alpha,\delta)}$. By previous claims, since (α, δ) belongs to the set of keys, $\mathbf{g}_s^{(\alpha,\delta)} = k_{s,\delta} \cdot h_{s,\xi^{-1}(\delta)}$. Therefore

$$h_{s,\gamma} \; = \; \mathbf{g}^{(\alpha,\delta)} \cdot k_{s,\delta}^{-1} \; = \; h_{s,\xi^{-1}(\delta)} \quad \Rightarrow \quad \gamma = \xi^{-1}(\delta)$$

where the implication follows by the fact that all the elements $h_{s,1}, \ldots, h_{s,n}$ are distinct. In conclusion $\xi^{-1}(\gamma) = \delta \;\Rightarrow\; (j, \xi^{-1}(\gamma)) = (j, \delta) \in R$ which implies by construction $(eid, j) \in E$.

Claim 7. $\mathsf{DDH} \;\Rightarrow\; \mathsf{H}_7 \equiv \mathsf{H}_6$. Given a distinguisher $\mathcal{Z}$ which perform at most U elections and V registrations we prove the statement through a sequence of hybrid games. We let $\mathsf{H}^*_{\sigma,d}$ be as H_6 but if an honest shuffle happens at round $s < \sigma$, perform it as in H_7 whereas if it occurs at round $s = \sigma$, perform it as in H_7 only for those ℓ linked to honest users such that $\xi(\ell) < d$. Notice that for $\mathcal{Z}$, $\mathsf{H}^*_{0,0} \equiv \mathsf{H}_6$, $\mathsf{H}^*_{\sigma,V} \equiv \mathsf{H}^*_{\sigma+1,0}$ and $\mathsf{H}^*_{U,V} \equiv \mathsf{H}_7$, so we only need to prove $\mathsf{H}^*_{\sigma,d} \equiv \mathsf{H}^*_{\sigma,d+1}$ by reducing it through an algorithm $\mathcal{C}$ to DDH.

Description of $\mathcal{C}$. On input $(g, w, \widetilde{g}, \widetilde{w})$ sample a trapdoor for the Pedersen commitment $\mathbf{y} \xleftarrow{\$} \mathbb{F}_q^2$, set $\mathbf{g} \leftarrow g^{\mathbf{y}}$, $\widetilde{\mathbf{g}} = \widetilde{g}^{\mathbf{y}}$, $\rho \xleftarrow{\$} 1$ and simulate $\mathsf{H}^*_{\sigma,d}$ with the following changes.

- $\mathcal{F}_{\mathsf{ct}}$ initially returns $\mathbf{g}_0 = \mathbf{g}$
- When a user P_j registers set $\mathbf{v}_{s,n} \leftarrow \mathbf{0}$. If P_j is corrupted extract $\mathbf{z}_{s,n} \in \mathbb{F}_q^2$ such that $h_{s,n} = \mathbf{g}_s^{\mathbf{z}_{s,n}}$. If P_j is honest and $n \neq d$ sample $\mathbf{z}_{s,n} \xleftarrow{\$} \mathbb{F}_q^2$ and set $h_{s,n} \leftarrow \mathbf{g}_s^{\mathbf{z}_{s,n}}$. Otherwise if $n = d$ set $h_{s,n} \leftarrow w$, $\vartheta_s \leftarrow 1$ and $\mu_s \leftarrow 0$.

– After any correct shuffle let (r_{s+1}, η_{s+1}) be the witness either used by an honest shuffle, performed by setting $r_{s+1} = 1$ if $s = \sigma$, or extracted from the NIZK proof of a corrupted shuffler. Update

$$\rho \leftarrow \rho \cdot r_{s+1} \qquad \xi \leftarrow \xi \circ \eta_{s+1}$$
$$\mathbf{z}_{s+1,\ell} \leftarrow \mathbf{z}_{s,\eta_{s+1}(\ell)} \qquad \mathbf{v}_{s+1,\delta} \leftarrow \mathbf{v}_{s,\delta} \qquad \vartheta_{s+1} \leftarrow \vartheta_s.$$

Moreover, if the shuffler is honest, simulate it according to the following cases:

- $s < \sigma$: For all $\ell \in [n]$ with $\xi(\ell) \neq d$ behave as in H_7 by setting $\alpha \in \mathbb{F}_q$ such that $\mathbf{y}^\top(\alpha, \delta) = \mathbf{y}^\top(\mathbf{z}_{s,\eta(\ell)} + \mathbf{v}_{s,\ell})$. For $\xi(\ell) = d$ set

$$\vartheta_{s+1} \leftarrow^{\$} \mathbb{F}_q, \qquad h_{s+1,\ell} \leftarrow w^{\rho\vartheta_{s+1}}, \qquad k_{s+1,\ell} \leftarrow h_{s+1,\ell}^{-1} \cdot \mathbf{g}_{s+1}^{(\mu_s,\delta)}$$

- $s = \sigma$: If the index d is linked to a malicious user, i.e. if $(j, d) \in R$ with $j \in M$, execute this shuffle as specified in $\mathsf{H}^*_{\sigma,d}$. Otherwise set $\mathbf{g}_{s+1} \leftarrow \widetilde{\mathbf{g}}^\rho$ and for all $\ell \in [n]$ linked to honest users with $\delta = \xi(\ell) < d$ compute $h_{s+1,\ell}, k_{s+1,\delta}$ as H_7 with the same procedure described before. For ℓ such that $\xi(\ell) = d$ set

$$h_{s+1,\ell} \leftarrow \widetilde{w}^{\rho\vartheta_{s+1}}, \qquad k_{s+1,\ell} \leftarrow \widetilde{w}^{-\rho\vartheta_{s+1}} \cdot \mathbf{g}_{s+1}^{(\mu_s,\delta)}$$

  For other values of ℓ instead set

$$h_{s+1,\ell} \leftarrow \mathbf{g}_{s+1}^{\mathbf{z}_{s+1,\ell}}, \qquad k_{s+1,\delta} \leftarrow \mathbf{g}_{s+1}^{\mathbf{v}_{s+1,\delta}}$$

- $s > \delta$ perform a correct shuffle as in the real protocol.

– After a shuffle has been performed simulate the update by extracting $\mathbf{w}_{s,\delta} \in \mathbb{F}_q^2$ such that $u_{s,\delta} = \mathbf{g}_s^{\mathbf{w}_{s,\delta}}$ from the proof of corrupted users and setting $\mathbf{v}_{s,\delta} \leftarrow \mathbf{v}_{s,\delta} + \mathbf{w}_{s,\delta}$. Instead for all $(j, \delta) \in R$ with P_j not corrupted and $\delta \neq d$

$$\mathbf{v}_{s,\delta} \leftarrow^{\$} \mathbb{F}_q^2, \qquad u_{s,\delta} \leftarrow k_{s,\delta}^{-1} \cdot \mathbf{g}_s^{\mathbf{v}_{s,\delta}}.$$

For $\delta = d$ instead, calling $\gamma = \xi^{-1}(\delta)$

$$\mu_s \leftarrow^{\$} \mathbb{F}_q, \qquad u_{s,\delta} \leftarrow h_{s,\gamma}^{-1} \cdot k_{s,\delta}^{-1} \cdot \mathbf{g}_s^{(\mu_s,\delta)}.$$

– When $\mathcal{Z}$ send $(\mathsf{corrupt}, j)$, for all $\delta \in R_j$ with $\delta \neq d$ find $\alpha \in \mathbb{F}_q$ such that, calling $\gamma = \xi^{-1}(\delta)$

$$\mathbf{y}^\top(\alpha, \delta) \ = \ \mathbf{y}^\top(\mathbf{z}_{s,\gamma} + \mathbf{v}_{s,\delta})$$

otherwise, for $\delta = d$ set $\alpha = \mu_s$. Add (α, δ) to $\mathcal{K}_{s,j}$ and send $(\mathsf{corrupted}, j, \mathcal{K}_{s,j})$

Finally, when $b \leftarrow^{\$} \mathcal{Z}$ and halts, return b and halt.

Proof of Claim. First of all we observe that one can easily show by induction the following properties for all s and ℓ calling $\delta = \xi(\ell)$:

$$\begin{aligned}
s < \sigma \quad &\Rightarrow \quad \mathbf{g}_s = \mathbf{g}^\rho \\
s \geq \sigma \quad &\Rightarrow \quad \mathbf{g}_s = \widetilde{\mathbf{g}}^\rho \\
\delta \neq d \quad &\Rightarrow \quad h_{s,\ell} = \mathbf{g}_s^{\mathbf{z}_{s,\ell}}, \quad k_{s,\delta} = \mathbf{g}_s^{\mathbf{v}_{s,\delta}} \\
s < \sigma,\ \delta = d \quad &\Rightarrow \quad h_{s,\ell} = w^{\rho\vartheta_s}, \quad k_{s,\delta} = w^{-\rho\vartheta_s}\mathbf{g}_s^{(\mu_s,\delta)} \\
s \geq \sigma,\ \delta = d \quad &\Rightarrow \quad h_{s,\ell} = \widetilde{w}^{\rho\vartheta_s}, \quad k_{s,\delta} = \widetilde{w}^{-\rho\vartheta_s}\mathbf{g}_s^{(\mu_s,\delta)}
\end{aligned}$$

Given this we proceed to show that regardless of the tuple $\mathcal{C}$ receives, it correctly reproduces the phases H^*_σ and $\mathsf{H}^*_{\sigma-1}$ agree on. Registrations are correctly simulated since with both functionalities honest parties would reply with $h_{s,n} = \mathbf{g}_s^{(\alpha,n)}$ for a uniform α meaning that $h_{s,n} \sim U(\mathbb{G})$ and independent from previous messages, as in the protocol simulated by $\mathcal{C}$ where $h_{s,n} = \mathbf{g}_s^{\mathbf{z}_{s,n}}$ or $h_{s,n} = w^\rho$. Updates are also correctly distributed since in both functionalities $u_{s,\delta} \sim U(\mathbb{G})$ and independent from other messages. In the view generated by $\mathcal{C}$, $\mathbf{g}_s^{\mathbf{v}_{s,\delta}}$ is uniform and independent, which implies that so is $u_{s,\delta} = k_{s,\delta}^{-1} \cdot \mathbf{g}_s^{\mathbf{v}_{s,\delta}}$ for $\delta \neq d$. The same holds for $\delta = d$ since $\mu_s \sim U(\mathbb{F}_q)$.

Next we show that corruption is handled correctly. For every $\delta \in R_j$ if $\delta \neq d$ and $s < \sigma$ then

$$h_{s,\ell} \cdot k_{s,\delta} \;=\; \mathbf{g}_s^{\mathbf{z}_{s,\ell}+\mathbf{v}_{s,\delta}} \;=\; g^{\rho\mathbf{y}^\top(\mathbf{z}_{s,\ell}+\mathbf{v}_{s,\delta})} \;=\; g^{\rho\mathbf{y}^\top(\alpha,\delta)} \;=\; \mathbf{g}_s^{(\alpha,\delta)}$$

while the case $s \geq \sigma$ is analogous. Conversely, if $\delta = d$ then $h_{s,\ell} \cdot k_{s,\delta} = \mathbf{g}_s^{(\mu_s,\delta)}$ which implies that the exponent H_7 would provide is μ_s.

Next we prove that $\mathcal{C}$ simulates correctly all the shuffle rounds for $s \neq \sigma$. When $s < \sigma$ and the shuffler is not corrupted we have that, with the same argument used above, the exponent α is the same H_7 would pick. Moreover for ℓ such that $\xi(\ell) = d$ we have $h_{s+1,\ell} \sim U(\mathbb{G})$ independently from previous message, and μ_s is the exponent required for the auxiliary term $k_{s+1,\ell}$ since $h_{s,\eta(\ell)} \cdot k_{s,\delta} = \mathbf{g}^{(\mu_s,\delta)}$. The case $s > \sigma$ is trivial.

Regarding the shuffle at round $s = \sigma$ instead we can prove that all the elements $h_{s+1,\ell}$, $k_{s+1,\delta}$ are correctly computed for $\xi(\ell) \neq d$. Indeed calling $r \in \mathbb{F}_q$ the exponent such that $\widetilde{g} = g^r$ we have $\widetilde{\mathbf{g}} = \mathbf{g}^r$ and in particular

$$\mathbf{g}_{s+1} \;=\; \widetilde{\mathbf{g}}^\rho \;=\; \mathbf{g}^{r\rho} \;=\; \mathbf{g}_s^r.$$

For all $\ell \in [n]$ linked to honest users and with $\xi(\ell) < d$, α is the same exponent H_7 would compute, for other values of ℓ with $\xi(\ell) \neq d$

$$\begin{aligned} h_{s+1,\ell} \;&=\; \mathbf{g}_{s+1}^{\mathbf{z}_{s+1,\ell}} \;=\; \mathbf{g}_s^{r\mathbf{z}_{s,\eta_{s+1}(\ell)}} \;=\; h^r_{s,\eta_{s+1}(\ell)} \\ k_{s+1,\delta} \;&=\; \mathbf{g}_{s+1}^{\mathbf{v}_{s+1,\delta}} \;=\; \mathbf{g}_{s+1}^{r\mathbf{v}_{s,\delta}} \;=\; k^r_{s,\delta}. \end{aligned}$$

Finally we will show that $\mathcal{C}$ in DDH^1 simulates $\mathsf{H}^*_{\sigma,d}$ while in DDH^0 emulates $\mathsf{H}^*_{\sigma,d+1}$. As a preliminary step we observe that if $\mathcal{Z}$ corrupts P_j with $(j,d) \in R$ before the σ-th shuffle then the views in $\mathsf{H}^*_{\sigma,d}$ and $\mathsf{H}^*_{\sigma,d+1}$ is identical and $\mathcal{C}$ correctly simulates both. Thus we will assume from now on that $\mathcal{C}$ does not corrupt P_j.

When $\mathcal{C}$ is executed in DDH^1, calling again $r \in \mathbb{F}_q$ the exponent such that $\widetilde{g} = g^r$, then $\widetilde{w} = w^r$. Therefore if the σ-th shuffler is honest, calling $\ell = \xi^{-1}(d)$

$$h_{s+1,\ell} \;=\; \widetilde{w}^{\rho\vartheta_{s+1}} \;=\; w^{r\rho\vartheta_s} \;=\; h^r_{s,\ell'}$$

for some $\ell' \in [n]$ where $\xi_s(\ell') = \delta = \xi_{s+1}(\ell)$ which implies $\ell' = \eta_{s+1}(\ell)$ and in particular $h_{s+1,\ell} = h^r_{s,\eta(\ell)}$ as prescribed in $\mathsf{H}^\star_{\sigma,d}$. Moreover

$$k_{s+1,\delta} \;=\; \widetilde{w}^{-\rho\vartheta_{s+1}} \cdot \mathbf{g}_{s+1}^{(\mu_s,\delta)} \;=\; w^{-r\rho\vartheta_s} \cdot \mathbf{g}_s^{r(\mu_s,\delta)} \;=\; k^r_{s,\delta}.$$

When $\mathcal{C}$ is executed in DDH^0 instead we have that $\widetilde{w}$ is uniformly and independently random, and in particular so is $h_{s+1,\ell} = \widetilde{w}^{\rho\vartheta_{s+1}}$. Finally μ_s is the exponent required to compute $k_{s+1,\delta}$ as specified in H_7. Indeed, according to the equation presented at the beginning, $h_{s,\eta(\ell)} \cdot k_{s,\delta} = \mathbf{g}_s^{(\mu_s,\mu_s\delta)}$, which proves that $k_{s+1,\ell}$ is computed as it would in $\mathsf{H}^*_{\sigma,d+1}$. The claim follows.

Claim 8. $\mathsf{H}_7 \equiv \mathsf{H}_8$: After any election let (γ_0, j_0) and (γ_1, j_1) be random variables such that, respectively in H_7 and H_8, $\mathcal{F}^n_{\mathsf{ct}}$ returns $(\mathsf{tossed}, eid, \gamma_b)$ and (eid, j_b) is added in E for $b \in \{0,1\}$. We will prove this two elements follow the same distribution. First of all we observe that the set R defines a function $f : [n] \to [N]$ such that $f(\delta) = j$ if and only if $(j, \delta) \in R$. This is true since for all $\delta \in [n]$ there is a party P_j who performed the δ-th registration and this user is unique. A trivial consequence of this definition is that $R_j = f^{-1}(j)$.

In H_7 by construction $\gamma_0 \sim U([n])$ chosen virtually by $\mathcal{F}^n_{\mathsf{ct}}$, $\delta_0 = \xi(\gamma_0)$ and with the above notation $j_0 = f(\delta_0) = f \circ \xi(\gamma_0)$. Conversely in H_8, $(j_1, \cdot) \sim U(R)$, $\delta_1 \sim U(R_{j_1})$ and $\gamma_1 = \xi^{-1}(\delta_1)$. Notice that since ξ^{-1} is a bijection $\gamma_1 \sim U([n])$ if and only if $\delta_1 \sim U([n])$. In order to show the latter, taken any $x \in [n]$

$$\begin{aligned}
\Pr[\delta_1 = x] &= \sum_{y \in [N]} \Pr[\delta_1 = x | j_1 = y] \Pr[j_1 = y] \\
&= \Pr[\delta_1 = x | j_1 = f(x)] \Pr[j_1 = f(x)] \\
&= \frac{1}{|f^{-1}(x)|} \cdot \frac{|f^{-1}(x)|}{N} = \frac{1}{N}
\end{aligned}$$

where the second equality holds since if $j_1 = y \neq f(x)$ then by construction $\delta_1 \in f^{-1}(y)$ but $x \notin f^{-1}(y)$ which implies $\delta_1 \neq x$. We obtain then $\gamma_1 \sim U([n])$.

To finally show that $\Delta((\gamma_0, j_0), (\gamma_1, j_1)) = 0$ observe that both γ_0, γ_1 are uniform and that upon conditioning on $\gamma_0 = z$ and $\gamma_1 = z$ we have $j_0 = f \circ \xi(z)$ in H_7 and $\delta_1 = \xi(z) \in R_{j_1} \Rightarrow j_1 = f \circ \xi(z)$ in H_8. By Proposition 3 the two distributions are therefore equivalent.

Claim 9. $\mathsf{DDH}_3 \Rightarrow \mathsf{H}_9 \equiv \mathsf{H}_8$: Let $\mathsf{H}^\star_\theta$ be an hybrid functionality such that $\mathsf{H}^\star_\theta$ perform a swap, as specified in H_9 for the first θ elections with an honest winner. Since $\mathsf{H}^\star_0 = \mathsf{H}_8$ and $\mathsf{H}^\star_U = \mathsf{H}_9$ when executed with $\mathcal{Z}$, it is enough to show $\mathsf{H}^\star_\theta = \mathsf{H}^\star_{\theta+1}$. Given a distinguished $\mathcal{Z}$ we describe $\mathcal{D}$ that breaks DDH_3 whose advantage is significant compared to the advantage of $\mathcal{Z}$.

Description of $\mathcal{D}$: On input $(g, w_0, w_1, \widetilde{g}, \widetilde{w}_0, \widetilde{w}_1)$, sample/setup the following

$$\begin{array}{llll}
\mathbf{y} \leftarrow^\$ \mathbb{F}_q^2, & \delta_0 \leftarrow^\$ [V], & \delta_1 \leftarrow^\$ [V], & b \leftarrow^\$ \{0,1\}, \\
\rho \leftarrow 1, & \mathsf{cnt} \leftarrow 0, & \mathsf{state} \leftarrow \mathsf{wait}, &
\end{array}$$

where cnt counts the past elections with an honest winner, $\mathsf{state} \in \{\mathsf{wait}, \mathsf{done}\}$ specifies if we injected the DDH_3 challenge yet and δ_0, δ_1 are guesses on the two honest party's indices that may be swapped after the $(\vartheta + 1)$-th election with an honest winner. Call $\mathbf{g} \leftarrow g^{\mathbf{y}}$, $\widetilde{\mathbf{g}} \leftarrow \widetilde{g}^{\mathbf{y}}$ and define IC, *injection condition*, the statement

$$(\,\cdot\,, \delta_0) \in R \quad \wedge \quad (\,\cdot\,, \delta_1) \in R \quad \wedge \quad \mathsf{cnt} = \theta.$$

Next simulate $\mathsf{H}^\star_\theta$ with the following changes:

– Initially let $\mathcal{F}_{\mathsf{ct}}$ return $\mathbf{g}_0 = \mathbf{g}$.
– When a party P_j performs a registration set $\mathbf{v}_{s,n} = 0$. If P_j is corrupted extract $\mathbf{z}_{s,n} \in \mathbb{F}_q^2$ such that $h_{s,n} = \mathbf{g}_s^{\mathbf{z}_{s,n}}$, otherwise sample $\mathbf{z}_{s,n} \leftarrow^\$ \mathbb{F}_q^2$ and set $h_{s,n} \leftarrow \mathbf{g}_s^{\mathbf{z}_{s,n}}$

– When a shuffle is performed by a corrupted user extract from the provided proof the witness r_{s+1}, η_{s+1}. Otherwise, if an honest user has to perform a shuffle, sample $r_{s+1} \xleftarrow{\$} \mathbb{F}_q$ and $\eta_{s+1} \xleftarrow{\$} S_n$. Update

$$\rho \leftarrow \rho \cdot r_{s+1}, \qquad \xi \leftarrow \xi \circ \eta_{s+1}, \qquad \mathbf{z}_{s+1,\ell} \leftarrow \mathbf{z}_{s,\eta_{s+1}(\ell)}, \qquad \mathbf{v}_{s+1,\delta} \leftarrow \mathbf{v}_{s,\delta}$$

and, only if the shuffler is honest, simulate it according to the following cases:

- $(\mathsf{state} = \mathsf{wait}) \wedge \neg\mathsf{IC}$. For all $\ell \in [n]$ linked to honest parties, sample $\mathbf{z}_{s+1,\ell} \xleftarrow{\$} \mathbb{F}_q^2$ and set $h_{s+1,\ell} \leftarrow \mathbf{g}_{s+1}^{\mathbf{z}_{s+1,\ell}}$. Next call $\delta = \xi(\ell)$ and compute

$$\alpha \in \mathbb{F}_q \ : \ \mathbf{y}^\top(\alpha, \delta) = \mathbf{y}^\top(\mathbf{z}_{s,\eta_{s+1}(\ell)} + \mathbf{v}_{s,\delta}), \qquad \mathbf{v}_{s+1,\delta} \leftarrow (\alpha, \delta) - \mathbf{z}_{s+1,\ell}.$$

  Finally set the auxiliary element for the key update as

$$k_{s+1,\delta} \ \leftarrow \ \begin{cases} \mathbf{g}_{s+1}^{\mathbf{v}_{s+1,\delta}} & \text{If } \delta \notin \{\delta_0, \delta_1\} \\ w_\beta^\rho \cdot \mathbf{g}_{s+1}^{\mathbf{v}_{s+1,\delta}} & \text{If } \delta = \delta_\beta, \ \beta \in \{0,1\} \end{cases}$$

- $(\mathsf{state} = \mathsf{wait}) \wedge \mathsf{IC}$. Set $\mathsf{state} = \mathsf{done}$, $r_{s+1} = 1$ and $\mathbf{g}_{s+1} \leftarrow \widetilde{\mathbf{g}}^\rho$. Next, for all $\ell \in [n]$, if ℓ is linked to an honest party set with the above notation

$$\mathbf{z}_{s+1,\ell} \xleftarrow{\$} \mathbb{F}_q^2, \qquad \mathbf{v}_{s+1,\delta} \xleftarrow{\$} \mathbf{z}_{s+1,\ell} - (\alpha, \delta), \qquad h_{s+1,\ell} \leftarrow \mathbf{g}_{s+1}^{\mathbf{z}_{s+1,\ell}}$$
$$k_{s+1,\delta} \leftarrow \begin{cases} \mathbf{g}_{s+1}^{\mathbf{v}_{s+1,\delta}} & \text{If } \delta \notin \{\delta_0, \delta_1\} \\ \widetilde{w}_\beta^\rho \cdot \mathbf{g}_{s+1}^{\mathbf{v}_{s+1,\delta}} & \text{If } \delta = \delta_\beta \ \beta \in \{0,1\} \end{cases}$$

  where $\alpha \in \mathbb{F}_q$ is computed as in the previous case. Else, if ℓ is linked to a corrupted player, set

$$h_{s+1,\ell} \leftarrow \mathbf{g}_{s+1}^{\mathbf{z}_{s+1,\ell}}, \qquad k_{s+1,\delta} \leftarrow \mathbf{g}_{s+1}^{\mathbf{v}_{s+1,\delta}}.$$

- If $\mathsf{state} = \mathsf{done}$, simulate the shuffle phase as prescribed in $\mathsf{H}_\theta^\star$.

– After any shuffle, simulate the update by extracting $\mathbf{w}_{s,\delta} \in \mathbb{F}_2$ such that $u_{s,\ell} = \mathbf{g}_s^{\mathbf{w}_{s,\delta}}$ for the proof of any corrupted party and setting $\mathbf{v}_{s,\delta} \leftarrow \mathbf{v}_{s,\delta} + \mathbf{w}_{s,\delta}$. Conversely, for all $\delta \in [n]$ linked to an honest player, sample $\mathbf{v}_{s+1,\delta} \xleftarrow{\$} \mathbb{F}_q^2$ and set

$$u_{s+1,\delta} \ = \ \begin{cases} k_{s,\delta}^{-1} \cdot w_\beta^\rho \cdot \mathbf{g}_s^{\mathbf{v}_{s,\delta}} & \text{If } (\mathsf{state} = \mathsf{wait}) \ \wedge \ \delta = \delta_\beta \\ k_{s,\delta}^{-1} \cdot \mathbf{g}_s^{\mathbf{v}_{s,\delta}} & \text{Otherwise} \end{cases}$$

– When users request the election eid, sample $(j, \cdot\,) \xleftarrow{\$} R$ and increment $\mathsf{cnt} \leftarrow \mathsf{cnt} + 1$ if P_j is honest. Next, if $\mathsf{cnt} \neq \theta$ perform the election as in $\mathsf{H}_9^\star$ by adding (eid, j) in E. Else, for $\beta \in \{0,1\}$ set $\gamma_\beta = \xi(\delta_\beta)$ and $j_\beta \in [N]$ such that $(j_\beta, \delta_\beta) \in R$. Then store (eid, j_0) in E and make $\mathcal{F}_{\mathsf{ct}}^n$ return $(\mathsf{tossed}, eid, \gamma_b)$. When the honest winner ξ so that

$$\xi(\gamma_0) = \delta_b, \qquad \xi(\gamma_1) = \delta_{1-b}$$

i.e. perform a swap only if $b = 1$.

In conclusion, if at any time P_j is corrupted when $\mathsf{cnt} < \theta$ with $(j, \delta_0) \in R$ or $(j, \delta_1) \in R$ or if during the $(\theta+1)$-th honest election $(\,\cdot\,, \delta_0) \notin R$ or $(\,\cdot\,, \delta_1) \notin R$, return a random bit and halt. When $b' \xleftarrow{\$} \mathcal{Z}$, return $b == b'$.

Proof of Claim. First of all we study the behaviour of $\mathcal{D}$ when it is executed in $\mathsf{DDH}_3[1]$, i.e. when there exists an $r \in \mathbb{F}_q$ such that $\widetilde{g} = g^r$, $\widetilde{w}_0 = w_0^r$ and $\widetilde{w}_1 = w_1^r$, proving it agrees with $\mathsf{H}^\star_{\theta+b}$. We do this conditioning on $\neg\mathsf{halt}$, where halt is the event "*$\mathcal{D}$ forcefully halts during the execution and returns a random bit*". This involves the following phases

- *Registration phase*: As observed in other reduction, when an honest user register in $\mathsf{H}^\star_\theta$ it sends $h_{s,n} = \mathbf{g}_s^{(\alpha,n)}$ with $\alpha \sim U(\mathbb{F}_q)$, see the definition of H_2. Hence $h_{s,n} \sim U(\mathbb{G})$ independently from previous messages. Analogously, $\mathcal{D}$ sets $h_{s,n} = \mathbf{g}_s^{\mathbf{z}_{s,n}}$ with $\mathbf{z}_{s,n} \sim U(\mathbb{F}_q^2)$ implying that $h_{s,n} \sim U(\mathbb{G})$.

- *Update phase*: In $\mathsf{H}^\star_\theta$ honest users set $u_{s,\delta} \sim U(\mathbb{G})$ with fresh randomness, see the definition of H_7. Similarly, $\mathcal{D}$ samples with fresh randomness $\mathbf{v}_{s,\delta} \sim U(\mathbb{F}_q^2)$ meaning that $\mathbf{g}_s^{\mathbf{v}_{s,\delta}} \sim U(\mathbb{G})$. As a consequence both $k_{s,\delta}^{-1} \cdot \mathbf{g}_s^{\mathbf{v}_{s,\delta}}$ and $k_{s,\delta}^{-1} \cdot w_\beta^\rho \cdot \mathbf{g}_s^{\mathbf{v}_{s,\delta}}$ are uniformly distributed over $\mathbb{G}$ and independently from previous messages.

- *Election phase*: When $\mathsf{cnt} \neq \theta$ or the selected j lies in M the set of corrupted parties, then the election is carried out as in H_θ which for $\mathsf{cnt} < \theta$ and $j \notin M$ involves a swap as in $\mathsf{H}_{\theta+1}$ while for $\mathsf{cnt} > \theta$ and $j \notin M$ no swap is performed, again as in $\mathsf{H}_{\theta+1}$. Finally, when $\mathsf{cnt} = \theta$ and an honest winner is going to be selected, since we are conditioning on $\neg\mathsf{halt}$ we have that $(j_0, \delta_0) \in R$ and $(j_1, \delta_1) \in R$ with $j_0, j_1 \notin M$. By construction $\delta_0 \sim U([V])$ implies, under the condition $\neg\mathsf{halt}$ that
$$\delta_0, \delta_1 \sim U(\{\delta \in [n] \;:\; (j, \delta) \in R, \quad j \notin M\}).$$
As done in Claim 8, is easy to observe that (j_0, δ_0) then has the same distribution of a uniformly chose $j^* \xleftarrow{\$} [N] \setminus M$ and δ^* such that $(j^*, \delta^*) \in R$. Finally if according to the value of b after the election $\mathcal{D}$ sets
$$\begin{aligned} b = 0 \quad &\Rightarrow \quad \xi(\gamma_0) \leftarrow \gamma_0 \;=\; \xi(\gamma_1) \leftarrow \gamma_1 \\ b = 1 \quad &\Rightarrow \quad \xi(\gamma_0) \leftarrow \gamma_1 \;=\; \xi(\gamma_1) \leftarrow \gamma_0 \end{aligned}$$
i.e. it correctly simulates this election as in $\mathsf{H}^\star_{\theta+b}$.

- *Shuffle phase*: We begin observing that by induction one can easily prove that until $\mathsf{state} = \mathsf{wait}$ then $\mathbf{g}_s = \mathbf{g}^\rho$, $h_{s,\ell} = \mathbf{g}_s^{\mathbf{z}_{s,\ell}}$ and
$$\delta \notin \{\delta_0, \delta_1\} \;\Rightarrow\; k_{s,\delta} = \mathbf{g}_s^{\mathbf{v}_{s,\delta}}, \qquad \delta = \delta_b \;\Rightarrow\; k_{s,\delta} = w_\beta^\rho \cdot \mathbf{g}_{s+1}^{\mathbf{v}_{s,\delta}} \tag{1}$$
If $\mathsf{state} = \mathsf{wait}$ and $\neg\mathsf{IC}$, to prove correctness we just need to show that the exponent α computed by $\mathcal{D}$ is correct for $\delta = \delta_\beta$. First we notice that up to negligible probability the first component of $\mathbf{g}_0$ is non-zero, and that conditioning to this event there exists a $\nu \in \mathbb{F}_q$ such that $w_\beta^{\rho_s} = \mathbf{g}_s^{(\omega,0)}$. On the other hand by construction $\mathbf{g}_s^{\mathbf{z}_{s,\ell}+\mathbf{v}_{s,\delta}} = \mathbf{g}_s^{(\alpha,\delta)}$. Hence we have that for $\delta = \delta_\beta$, $\beta \in \{0,1\}$
$$h_{s,\ell} \cdot k_{s,\delta} \;=\; \mathbf{g}_s^{\mathbf{z}_{s,\ell}} \cdot \mathbf{g}_s^{(\nu,0)} \cdot \mathbf{g}_s^{\mathbf{v}_{s,\ell}} \;=\; \mathbf{g}_s^{(\alpha+\nu,\delta)}.$$

Since $\mathcal{D}$ sets $\mathbf{v}_{s+1,\delta} = (\alpha, \delta) - \mathbf{z}_{s+1,\ell'}$ with $\ell = \eta(\ell')$ we have that

$$
\begin{aligned}
h_{s+1,\ell'} \cdot k_{s+1,\delta} \;=\; w_\beta^{\rho_{s+1}} \cdot \mathbf{g}_{s+1}^{(\alpha,\delta)} \;=\; w_\beta^{r_{s+1}\rho_s} \cdot \mathbf{g}_{s+1}^{(\alpha,\delta)} \;= \\
= \mathbf{g}_s^{r_{s+1}(\nu,0)} \cdot \mathbf{g}_{s+1}^{(\alpha,\delta)} \;=\; \mathbf{g}_{s+1}^{(\nu,0)} \cdot \mathbf{g}_{s+1}^{(\alpha,\delta)} \;=\; \mathbf{g}_{s+1}^{(\alpha+\nu,\delta)}.
\end{aligned}
$$

For $\delta \notin \{\delta_0, \delta_1\}$ the proof is analogous to the one in Claim 7. In the sequent case, if $\mathsf{state} = \mathsf{wait}$ and IC, using equation 1 we have that $\mathbf{g}_{s+1} = \widetilde{\mathbf{g}}^\rho = \mathbf{g}^{r\rho} = \mathbf{g}_s^r$. Consequently, for all $\ell \in [n]$, if ℓ is linked to a dishonest party

$$
h_{s+1,\ell} \;=\; \mathbf{g}_{s+1}^{\mathbf{z}_{s,\eta(\ell)}} \;=\; \mathbf{g}_s^{r\mathbf{z}_{s,\eta(\ell)}} \;=\; h_{s,\eta(\ell)}^r.
$$

and similarly $k_{s+1,\delta} = k_{s,\delta}^r$. If ℓ is linked to an honest party with $\xi(\ell) = \delta \notin \{\delta_0, \delta_1\}$ then $h_{s+1,\ell}$ and $k_{s+1,\delta}$ are correctly computed by construction. Finally when $\delta = \delta_\beta$, $h_{s+1,\ell}$ is still correctly computed and

$$
k_{s+1,\delta} \;=\; \widetilde{w}_\beta^\rho \cdot \mathbf{g}_{s+1}^{\mathbf{v}_{s,\delta}} \;=\; w_\beta^{r\rho} \cdot \mathbf{g}_s^{r\mathbf{v}_{s,\delta}}
$$

hence correctness follows as in the previous point. Finally when $\mathsf{state} = \mathsf{done}$ shuffles are done correctly by construction.

– *Corruption*: Corruption happens as in $\mathsf{H}_\theta^\star$ or $\mathsf{H}_{\theta+1}^\star$ equivocating the Pedersen commitment. Notice this cannot be done if $\mathsf{cnt} < \theta$ and P_j with $(j, \delta_0) \in R$ or $(j, \delta_1) \in R$ is corrupted, because in that case the exponent of w_β is not known. However in this case $\mathcal{D}$ aborts returning a random bit.

Summing up we showed that when the input is a DDH_3^1 tuple, $\mathcal{D}$ simulates, under the condition $\neg\mathsf{halt}$, $\mathsf{H}_\theta^\star$ if $b = 0$ or $\mathsf{H}_{\theta+1}^\star$ if $b = 1$. Finally we argue that either $\neg\mathsf{halt}$ happens with significant probability or $\mathcal{Z}$'s advantage is negligible. To this aim let bad be the event "*when the $(\theta + 1)$-th honest winner claims victory or it is corrupted, there is only one $\delta \in [n]$ linked to honest players*". Clearly if bad happens the eventual swap performed after the claim/corruption leaves ξ unchanged, meaning that it is information-theoretically hard for $\mathcal{Z}$ to distinguish the two functionality. In symbols $\mathsf{Adv}(\mathcal{Z}|\mathsf{bad}) = 0$.

Conversely, if $\neg\mathsf{bad}$, when the $(\theta+1)$-th honest winner is revealed/corrupted, there exist $\delta_0^*, \delta_1^* \in [n]$ linked to honest users such that δ_0 is associated to the winner. Since the simulation performed by $\mathcal{C}$ hides information-theoretically δ_0, δ_1 we have that δ_β and δ_β^* are independent and in particular $\Pr\left[\delta_\beta = \delta_\beta^* | \neg\mathsf{bad}\right] = V^{-2}$. Finally notice that $(\delta_\beta = \delta_\beta^*) \wedge \neg\mathsf{bad} \iff \neg\mathsf{halt}$. This allows us to compute

$$
\begin{aligned}
\mathsf{Adv}(\mathcal{Z}) &= \mathsf{Adv}(\mathcal{Z}|\mathsf{bad}) \Pr[\mathsf{bad}] + \mathsf{Adv}(\mathcal{Z}|\neg\mathsf{bad}) \Pr[\neg\mathsf{bad}] \\
&= \mathsf{Adv}(\mathcal{Z}|\neg\mathsf{bad}) \Pr[\neg\mathsf{bad}] \\
&= \Pr\left[\mathcal{Z}^{\mathsf{H}_{\theta+b}^\star} \to b \middle| \neg\mathsf{bad}\right] \Pr[\neg\mathsf{bad}] - \frac{1}{2} \cdot \Pr[\neg\mathsf{bad}].
\end{aligned}
$$

Moreover $\Pr[\neg\mathsf{halt}] = \Pr\left[(\delta_\beta = \delta^*_\beta),\ \neg\mathsf{bad}\right] = V^{-2}\Pr[\neg\mathsf{bad}]$ which implies (denoting $\mathcal{D}^1$ the algorithm $\mathcal{D}$ executed with DDH^1_3)

$$\begin{aligned}
\Pr\left[\mathcal{D}^1 \to 1\right] &= \Pr\left[\mathcal{D}^1 \to 1|\mathsf{halt}\right]\Pr[\mathsf{halt}] + \Pr\left[\mathcal{D}^1 \to 1|\neg\mathsf{halt}\right]\Pr[\neg\mathsf{halt}] \\
&= \frac{\Pr[\mathsf{halt}]}{2} + \Pr\left[\mathcal{Z}^{\mathsf{H}^\star_{\theta+b}} \to b\middle|\,(\delta_\beta = \delta^*_\beta),\ \neg\mathsf{bad}\right]\cdot\frac{\Pr[\neg\mathsf{bad}]}{V^2} \\
&= \frac{1}{2} - \frac{\Pr[\neg\mathsf{bad}]}{2V^2} + \frac{1}{V^2}\Pr\left[\mathcal{Z}^{\mathsf{H}^\star_{\theta+b}\to b}\middle|\neg\mathsf{bad}\right]\cdot\frac{\Pr[\neg\mathsf{bad}]}{V^2} \\
&= \frac{1}{2} + \frac{1}{V^2}\cdot\mathsf{Adv}(\mathcal{Z})\,.
\end{aligned}$$

Next we study the behaviour of $\mathcal{D}$ when it is executed in DDH^0_3, where $\widetilde{w}_0$ and $\widetilde{w}_1$ are random elements. Assuming that the injection occurs at round σ, i.e. in the σ-th shuffle, calling $\gamma_0 = \xi_\sigma^{-1}(\delta_0)$ and $\gamma_1 = \xi_\sigma^{-1}(\delta_1)$ we have that after the shuffle (with permutation $\eta_{s+1} = \eta$) the elements $h_{\sigma+1,\eta^{-1}(\ell_0)}, k_{\sigma+1,\delta_0}, h_{\sigma+1,\eta^{-1}(\ell_1)}$ and $k_{\sigma+1,\delta_1}$ are uniformly random. Since no other element contains information on $\eta^{-1}(\gamma_0)$ and $\eta^{-1}(\gamma_1)$, even knowing the output of η in all other points, something that an adversary with unbounded computational power can do, it is impossible to distinguish between this execution and another one in which $\widetilde{\eta}_{\sigma+1} = \eta\circ(\gamma_1,\gamma_0)$ where (γ_1,γ_0) is the transposition that switch this two elements.
In particular, if $\neg\mathsf{halt}$ occurs, i.e. if the two parties are not corrupted until the $(\theta+1)$-th election, the adversary cannot detect if a swap is applied between this two elements. As a consequence $\mathcal{Z}$ has no information on b an $\Pr[b' = b] = 1/2$, hence

$$\begin{aligned}
\Pr\left[\mathcal{D}^0 \to 1\right] &= \Pr\left[\mathcal{D}^0 \to 1|\mathsf{halt}\right]\Pr[\mathsf{halt}] + \Pr\left[\mathcal{D}^0 \to 1|\neg\mathsf{halt}\right]\Pr[\neg\mathsf{halt}] \\
&= \frac{\Pr[\mathsf{halt}]}{2} + \frac{\Pr[\neg\mathsf{halt}]}{2} = \frac{1}{2}.
\end{aligned}$$

We can finally conclude that $\mathsf{Adv}(\mathcal{D}) = V^{-2}\cdot\mathsf{Adv}(\mathcal{Z})$ which proves the claim.

Claim 10. $\mathsf{H}_9 \equiv \mathsf{H}_{10}$: follows by inspection.