Efficient and Universally Composable Single Secret Leader Election from Pairings

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

[1] Dipartimento di Matematica e Informatica, Università di Catania, Italy.
catalano@dmi.unict.it
[2] IMDEA Software Institute, Madrid, Spain.
{dario.fiore,emanuele.giunta}@imdea.org
[3] Scuola Superiore di Catania, Italy.

Abstract. Single Secret Leader Election (SSLE) protocols allow a set of users to elect a leader among them so that the identity of the winner remains secret until she decides to reveal herself. This notion was formalized and implemented in a recent result by Boneh, *et al.* (ACM Advances on Financial Technology 2020) and finds important applications in the area of Proof of Stake blockchains.
In this paper we propose new solutions to the problem that advance the state of the art both from a theoretical and a practical perspective. On the theoretical front, we propose a definition of SSLE in the universal composability framework. We believe this to be the right setting for highly concurrent contexts such as those of many blockchain-related applications. Next, we propose a UC-realization of SSLE from public key encryption with keyword search (PEKS) and based on the ability of distributing the PEKS key generation and encryption algorithms. Finally, we present an efficient MPC-friendly PEKS that allows us to efficiently instantiate the abstract scheme.
Our concrete construction compares favorably with previous work (both in terms of computational costs and in terms of overall communication overhead) while guaranteeing much stronger composability guarantees.

Table of Contents

1 Introduction

Leader Election protocols are of fundamental importance to realize consensus in distributed systems. The rise of blockchain and its numerous applications brought renewed interest on this topic and motivated the need to consider consensus protocols that also provide some secrecy guarantees. This is the case, for example, of leader elections in the context Proof of Stake blockchains (e.g., [AMM18, GHM$^+$17, KKKZ19, GOT19]) where one may wish to randomly select a *secret* leader, i.e., a leader that remains hidden until she reveals herself. In these contexts, leader-secrecy allows to protect against several attacks that would otherwise compromise the liveness of the blockchain. Indeed, if a malicious party could know the identity of a future leader, he could try to deny the leader's access to the network (using a denial of service attack, for instance) before the latter publishes her block, and this would affect, at least temporarily, the liveness and finality of the system. An adversary could also try to bribe a potential leader to influence the set of transactions that are going to be published.

Many existing solutions address this question by selecting a few potential leaders in *expectation* (e.g. [BGM16, BPS16]). This means that, for every given round, on expectation a single block leader is elected. Unfortunately, however, this also means that even many (or zero) leaders can be elected in any round.

This state of affairs led to the quest for an election protocol that secretly produces a *single* leader [Lab19], i.e., where *exactly* one single candidate is able to prove that she won the election. We note here that, although in principle this problem could be solved using general multiparty computation, the need of an efficient solution rules this option out.

The question was formally addressed in a recent work of Boneh *et al.* [BEHG20] who put forward the notion of *Single Secret Leader Election* (SSLE, from now on). Informally, an SSLE scheme is a distributed protocol that secretly elects a leader and satisfies *uniqueness* (exactly one leader is elected), *fairness* (all participants have the same probability of becoming the leader) and *unpredictability* (if the adversary does not win the election, she should not be able to guess the leader better than at random). Boneh *et al.* [BEHG20] also proposed three constructions meeting this notion that are based on different approaches and that achieve different efficiency (and security) tradeoffs (cf. Table 1 for a summary). Their first SSLE scheme relies on indistinguishability obfuscation (iO) [GGH$^+$13] and its main advantage is that every election involves a single constant-size message from the winner. At the same time, given the status of iO realizations, this SSLE protocol is of very limited (if any) practical interest. The second construction in [BEHG20] builds on Threshold Fully homomorphic Encryption (TFHE) [BGG$^+$18] and is asymptotically less efficient than the iO-based one: every election needs $O(t)$ communication (where t is a bound on the number of malicious users tolerated by the system) to partially decrypt a publicly computable ciphertext; after this, the winner can prove her victory. A nice aspect of the TFHE-based solution is that it actually requires only a *leveled* scheme for circuits that for, say, $N = 2^{16}$ participants, can be of depth as little as 10. On the other hand, other aspects of this solution makes it far from practical, e.g., it requires large $O(N \log N)$ secret key shares, and no concrete distributed setup (for the TFHE scheme) is explicitly provided in [BGG$^+$18]. So to the best of our knowledge one would have to rely on general multiparty computation techniques to achieve it. The third SSLE construction in [BEHG20] is based on shuffling and the decisional Diffie-Hellman assumption. Asymptotically, it performs worse than the other two solutions: every new election requires a freshly shuffled list

			Election	
SSLE	Setup	State	Comm.	Rounds
iO	trusted	$O(N)$	$O(1)$	0
TFHE	trusted	$O(N)$	$O(t)$	1
Shuffle-N	–	$O(N)$	$O(N)$	1
Shuffle-$\sqrt{N}$	–	$O(N)$	$O(\sqrt{N})$	1
Ours (worst)	trusted	$O(\lambda)$	$O(t)$	$\kappa + 3$
Ours (optim.)	trusted	$O(\lambda)$	$O(\kappa \log N)$	4

Table 1: Comparison between the SSLE solutions from [BEHG20] and the SSLE of this work. In our case, κ is a statistical security parameter, which gives good enough security already for $\kappa \approx \log N$. Shuffle-$\sqrt{N}$ achieves a weak unpredictability notion.

of N Diffie-Hellman pairs[4] (along with a NIZK of shuffle). The authors also describe a lightweight variant whose communication costs are $O(\sqrt{N})$, but the tradeoff here is a scheme with significantly lower security guarantees, as the secret leader is selected in a public subset of only $\sqrt{N}$ users.

We note that all the three SSLE schemes of [BEHG20] require users to maintain an $O(N)$-long state information in order to perform and verify elections, and that the iO- and TFHE-based SSLE need a trusted setup. The latter must be realized with a distributed protocol and should be in principle refreshed when new users join the system. On the other hand, the shuffle-based solution is essentially setup-free and thus can handle more easily users that join and leave the system dynamically.

Beyond efficiency considerations, another fundamental limitation of the constructions above is that they are proved secure with respect to a (stand-alone) game-based definition which makes their actual security in concurrent settings unclear. This is problematic in practice as it is hardly the case that distributed consensus protocols are executed stand-alone.

Given this state of affairs, the main question that motivates our work is:
is it possible to build an SSLE protocol where the communication costs of an election are sub-linear in the number of players and that achieves good practical performances while also realizing strong composability guarantees?

1.1 Our contribution

In this paper we propose a new SSLE solution that answers the above question in the affirmative. Our first contribution is the proposal of a new definition of SSLE in the universal composability model [Can01] (see Section 3). We believe this to be the right notion to model security in the highly distributed, often concurrent, blockchain-like applications where electing a leader is required. Our new definition implies the game-based definition of Boneh *et al.* [BEHG20], but, needless to say, the converse is not true.

As a second contribution, we propose a UC-secure construction of SSLE. Our protocol is based on public key encryption with keyword search (PEKS) [BDOP04] and performs an election with communication $O(\kappa \log N)$ in an optimistic scenario where a subset of users correctly collaborate in

[4] Precisely, when the winner no longer wants to participate in future elections, there is no need to shuffle for the next election; we ignore this special case in our analysis.

the protocol (where κ is a statistical security parameter that can be as low as 10), and $O(t)$ in the worst case.

An overview of our SSLE protocol. Let us describe our protocol and its efficiency in slightly more detail. PEKS is a notion of functional encryption [BSW11, O'N10] in which given a ciphertext c encrypting a keyword w and secret key sk associated to another keyword w', the decryption allows one to learn if $w = w'$ and nothing more. Our SSLE protocol is based on the simple idea that for every election a small committee of users generates a ciphertext c that encrypts a random keyword $j \in \{1, \dots, N\}$, every user is given a secret key sk_i associated to an integer i, and can claim victory by giving a NIZK proof that she can decrypt the election's ciphertext.

More specifically, our protocol consists of three phases: (1) a setup in which the users run an MPC protocol to generate the public key of the PEKS and distribute its secret keys, (2) an election's preprocessing in which a randomly sampled committee of κ players generates a commitment to the election's ciphertext in a distributed way, and (3) the election online phase in which the commitment is opened by the committee and the winner claims victory.

We formalize this approach in a generic SSLE protocol that we prove UC-secure assuming ideal functionalities for the setup and encryption algorithms of any PEKS (see Section 4). Our main technical contribution, however is to design an efficient instantiation of this blueprint, by showing an "MPC-friendly" PEKS and by proposing efficient protocols for the setup and election phases, which correspond to distributed versions of the PEKS key generation and encryption algorithms respectively. To devise such a PEKS we build on (a modified variant of) the functional encryption for orthogonality (OFE) scheme recently proposed by Wee [Wee17] and on a novel technique so that the PEKS can test keywords equality $\mod N$ albeit the message space is over a large field $\mathbb{F}_q$. We refer to Section 5.1 for an informal overview of our techniques and the technical challenges to be overcome.

Analysis and comparison with previous SSLE protocols. In an optimistic scenario where all the committee members send correct messages, our election's preprocessing completes in 3 rounds in which every player in the committee sends $O(\log N)$ group elements. In the worst case, preprocessing can additionally take about κ rounds in which, roughly speaking, a faulty player is excluded and the protocol is restarted. The online phase instead requires one message from every committee member to open the committed ciphertext. In the bad case that some player does not collaborate in this opening, the protocol resorts to a threshold decryption in which t players send partial decryptions of the commitment, where t is the minimum number of honest players that is assumed (e.g., $t = N/2 + 1$).

In summary, our SSLE protocol achieves, in the worst case, similar asymptotic efficiency as the TFHE-based construction of [BEHG20] (a setup protocol for N players, and an $O(t)$ election's cost), but significantly better concrete costs thanks to our efficient realization. In the optimistic scenario, our protocol performs an election with a communication of $O(\kappa \log N)$ group elements, where κ can be as low as $\log N$. For instance, for an instantiation with $N = 2^{14}$ participants (as suggested in [Lab19]), the election requires a total of 47 kB in the offline phase and 2 kB in the online one. A downside of our protocol compared to the TFHE-based one[5] is that we need the additional 3 rounds of preprocessing. However we stress that these can be performed offline, e.g., while performing other elections. Furthermore, and perhaps more interestingly, our preprocessing can be used to prepare *in*

[5] The TFHE solution of [BEHG20] requires one round of partial decryption followed by another one to claim victory.

parallel a batch of any size of election's ciphertexts. This way these 3 rounds can be easily amortized over many elections.

Compared to the shuffle-based solution of [BEHG20], our election's protocol requires much less communication in the optimistic scenario: $O(\kappa \log N)$ offline and $O(\kappa)$ online vs. $O(N)$ in [BEHG20]. This comes at the price of a somewhat heavy setup to generate the secret keys of all the users which, despite doable in 5 rounds, requires players to send roughly 6 kB to every other user. If we compare the total communication cost for 1000 elections, our protocol (including the setup) is still competitive as it entails a total communication sent/received by each player of 80 MB, in contrast to 1.1 GB in [BEHG20]. On another note, in our case the registration of new users is more convoluted as we need to perform a distributed key generation whenever a new user wants to join. To mitigate the problem, we propose a locally-static/globally-dynamic approach as follows. Informally, we assume that the life of the system is divided in epochs. In each epoch, the system is assumed to be static, parties can leave but no new user can join. We let new users join the system by including them in the set of participants for a *future* time period. This way, some (say λ), randomly chosen, users of current epoch T can setup the system for users of subsequent time periods without explicitly requiring an extra setup phase.

In summary, the main contribution of our solution is to propose the first efficient SSLE protocol with sub-linear election costs. We believe that realizing a practical UC-secure SSLE with a setup-free and dynamic registration and sub-linear election's costs is an interesting problem left for future work.

A word on the optimistic scenario. Finally, we would like to justify why it can be reasonable to assume that our protocol is executed in the optimistic scenario. An important feature of our protocol is that, when a malicious behavior occurs, we can identify misbehaving parties exactly. This allows us to consider the possibility of imposing financial penalties to bad players. Such punishments would then incentivize honest behavior when assuming that players are rational (and penalties are sufficiently high).

This approach was put forward in [ADMM14a] and [BK14] and it, essentially, requires participants to deposit (on the blockchain) a collateral associated to their key. If a party is caught cheating, the collateral is then distributed among the (honest) ones. Such countermeasures make sense in the contexts where SSLE is currently needed (e.g. Proof of stake blockchains) and make the assumption of honest behavior more acceptable in practice than in usual MPC scenarios.

1.2 Other related work

The problem of extending proof of stake systems to consider privacy was considered, among others, in [GOT19] and in [KKKZ19]. Leader election protocols were also considered by Algorand [GHM$^+$17] and Fantomette [AMM18]. There the idea is to first identify few potential leaders (via a VRF) that then reveal themselves in order and choose the winner via some simple tie break method (e.g. lowest VRF output wins). The approach is efficient but has the drawback that the elected leader does not know she was elected until everybody else published their value. Moreover, implicitly requires all nodes to be able to see the winner's output: users not getting this information might incorrectly think that another leader was elected (causing the chain to fork). We stress that this cannot happen in our setting.

The idea of imposing financial punishments to cheating users was also considered, generalized and improved (mainly in terms of amount of interaction required) in [KMB15, KB16, KZZ16]. In

[ADMM14a, ADMM14b, KMB15] authors considered the notion of MPC with cash distribution, where inputs and outputs of players consists of both data and money. Finally, Baum *et al.* [BDD20] recently proposed a modular approach to the problem called Insured MPC.

2 Preliminaries

2.1 Notation

Throughout this paper $\lambda \in \mathbb{N}$ is the main security parameter, and we say that a function $\varepsilon(\lambda)$ is *negligible* if it vanishes faster than the inverse of any polynomial in λ. We also use κ for a statistical security parameter. We denote $[n] = \{0, \ldots, n-1\}$. Bold font $(\mathbf{a}, \mathbf{u}, \mathbf{w}, \ldots)$ is reserved for vectors of field or group elements, while calligraphic font for Turing machines $(\mathcal{A}, \mathcal{B}, \mathcal{D}, \ldots)$. $x \leftarrow^{\$} S$ means that x is sampled uniformly and with fresh randomness from S. N is the number of players and t the threshold parameter.

We denote with $\mathcal{G}(1^\lambda)$ a *bilinear group generator*, that is an algorithm which takes as input the security parameter and outputs the description of a bilinear group setting $\mathsf{bg} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are groups of the same prime order $q > 2^\lambda$, $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ are two generators, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable, non-degenerate, bilinear map. We use $g_T = e(g_1, g_2)$ to denote a canonical generator of $\mathbb{G}_T$. When $\mathbb{G}_1 = \mathbb{G}_2$, the groups are called *symmetric*; otherwise they are called *asymmetric*. In our work we use Type-III *asymmetric* bilinear groups [GPS08] in which there is no known efficiently computable isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$.

We use $\mathbb{F}_q$ to denote the finite field of prime cardinality q. Given a vector $\mathbf{a} = (a_i)_{i=1}^n \in \mathbb{F}_q^n$ and a group element g we denote $[\mathbf{a}]_g = (g^{a_1}, \ldots, g^{a_n})$. When the base is g_1, g_2 or g_T we replace the above notation with $[\mathbf{a}]_1$, $[\mathbf{a}]_2$ and $[\mathbf{a}]_T$ respectively. Operations with vectors in $\mathbb{G}^n$ are entry-wise, i.e., for $\mathbf{g}, \mathbf{h} \in \mathbb{G}^n$, $\mathbf{g} \cdot \mathbf{h} = (g_i \cdot h_i)_{i=1}^n$, $\mathbf{g}^a = (g_i^a)_{i=1}^n$. Pairings are the only exception where $e(\mathbf{g}, \mathbf{h}) = e(g_1, h_1) \cdots e(g_n, h_n)$ for $\mathbf{g} \in \mathbb{G}_1^n$ and $\mathbf{h} \in \mathbb{G}_2^n$. Similarly $\mathbf{g}^{\mathbf{a}} = g_1^{a_1} \cdot \ldots \cdot g_n^{a_n}$.

2.2 SXDH assumption

In our efficient construction we rely on the SXDH assumption in bilinear groups, which informally states that the classical DDH assumption holds in both $\mathbb{G}_1$ and $\mathbb{G}_2$. More formally,

Definition 1 (SXDH **assumption).** *Let $\mathcal{G}$ be a bilinear group generator. We say that the* SXDH *assumption holds for $\mathcal{G}$ if for every PPT adversary $\mathcal{A}$, and every $s \in \{1, 2\}$ there exists a negligible function ε such that:*

$$|\Pr[\mathcal{A}(\mathsf{bg}, [a]_s, [b]_s, [c]_s) = 1] - \Pr[\mathcal{A}(\mathsf{bg}, [a]_s, [b]_s, [ab]_s) = 1]| \leq \varepsilon(\lambda)$$

where the probabilities are over the random choice of $a, b, c \leftarrow^{\$} \mathbb{F}_q$ and $\mathsf{bg} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2) \leftarrow^{\$} \mathcal{G}(1^\lambda)$.

When the above assumption is considered in only one group $\mathbb{G}_s$, for either $s = 1$ or $s = 2$, we refer to it as DDH in $\mathbb{G}_s$. We call DDH^0 a game in which $\mathcal{A}$ received the first distribution and DDH^1 a game in which he receives the second one. In the paper we also use an extension of DDH for vectors of n elements, called DDH_n, which briefly says that it is hard to distinguish the tuple $([a_1]_s, \ldots, [a_n]_s, [b]_s, [c_1]_s, \ldots, [c_n]_s)$, denoted as DDH_n^0, from the tuple $([a_1]_s, \ldots, [a_n]_s, [b]_s, [a_1 b]_s, \ldots, [a_n b]_s)$ denoted as DDH_n^1, for random $a_i, b, c_i \in \mathbb{F}_q$. We note that DDH_n can be reduced to DDH in the same group [NR97].

2.3 Functional Encryption

We recall the definition of Functional Encryption [BSW11, O'N10].

Definition 2. *A* functionality F *is a family of functions* $\mathsf{F} = \{f : \mathcal{X} \to \mathcal{Y}\}$*, where* $\mathcal{X}$ *is the plaintext space and* $\mathcal{Y}$ *is the output space.*

Definition 3. *A* ***functional encryption scheme*** *for a functionality* F *is a tuple* $(\mathsf{FE.Setup}, \mathsf{FE.Enc}, \mathsf{FE.KeyGen}, \mathsf{FE.Dec})$ *of* PPT *algorithms such that*

- $\mathsf{FE.Setup}(1^\lambda) \xrightarrow{\$} (\mathsf{mpk}, \mathsf{msk})$ *generates the secret and public master keys.*
- $\mathsf{FE.Enc}(m, \mathsf{mpk}; r) \to c$ *returns a ciphertext. The randomness* r *may be omitted.*
- $\mathsf{FE.KeyGen}(f, \mathsf{msk}) \xrightarrow{\$} \mathsf{sk}_f$ *returns a key associated to the function* $f \in \mathsf{F}$.
- $\mathsf{FE.Dec}(c, f, \mathsf{mpk}, \mathsf{sk}_f) \to x$ *a bit string.*

The scheme is ***correct*** *if for any* $m \in \mathcal{X}$ *and* $f \in \mathsf{F}$*, sampled* $\mathsf{mpk}, \mathsf{msk} \xleftarrow{\$} \mathsf{FE.Setup}(1^\lambda)$*,* $c \xleftarrow{\$} \mathsf{FE.Enc}(m, \mathsf{mpk})$*,* $\mathsf{sk}_f \xleftarrow{\$} \mathsf{FE.KeyGen}_{\mathsf{msk}}(f)$*, then up to negligible probability* $\mathsf{FE.Dec}(c, f, \mathsf{mpk}, \mathsf{sk}_f) = f(m)$.

We recall the definition of selective security for FE, which is sufficient for our application.

Definition 4. *A functional encryption scheme achieves* ***selective security*** *if for any* PPT *algorithm* $\mathcal{A}$ *there exists a negligible function* ε *such that*

$$\mathsf{Adv}^{\mathcal{A}}_{\mathsf{SSFE}}(1^\lambda) = \left| \Pr\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{SSFE}}(1^\lambda) = 1 \right] - \frac{1}{2} \right| \le \varepsilon(\lambda).$$

Selective Security Game $\mathsf{Exp}^{\mathcal{A}}_{\mathsf{SSFE}}(1^\lambda)$:

```
1 : $m_0, m_1 \xleftarrow{\$} \mathcal{A}$
2 : Sample $b \xleftarrow{\$} \{0,1\}$, $\mathsf{mpk}, \mathsf{msk} \xleftarrow{\$} \mathsf{FE.Setup}(1^\lambda)$, $c \xleftarrow{\$} \mathsf{FE.Enc}(m_b, \mathsf{mpk})$
3 : Send $\mathcal{A} \leftarrow \mathsf{mpk}, c$
4 : **When** $\mathcal{A}$ queries $f$, if $f(m_0) \neq f(m_1)$ then $\mathcal{A} \leftarrow \perp$. Otherwise:
5 :   Compute $\mathsf{sk}_j \xleftarrow{\$} \mathsf{FE.KeyGen}(f, \mathsf{msk})$ and send $\mathcal{A} \leftarrow \mathsf{sk}_f$
6 : **When** $\mathcal{A} \to b'$: Return $b \overset{?}{=} b'$
```

2.4 Functional Encryption for Keyword Search

In our work we make use of an FE scheme for the keyword search functionality [BDOP04, ABC+05] $\mathsf{F}_{ks} = \{f_y : \mathcal{X} \to \{0,1\}\}$, where each function $f_y \in \mathsf{F}_{ks}$ is defined by an element $y \in \mathcal{X}$ and is such that $f_y(x)$ returns 1 if $x = y$ and 0 otherwise. More precisely, our realization works with the more general functionality $\mathsf{F}^{B,N}_{ks} = \{f_y : [B] \to \{0,1\}\}$ parametrized by two positive integers B, N (of polynomial size), and such that each function $f_y \in \mathsf{F}^{B,N}_{ks}$ is defined by an integer $y \in [N]$ and $f_y(x)$ returns 1 if $x = y \bmod N$, and 0 otherwise. Note that F_{ks} for $\mathcal{X} = \mathbb{F}_q$ coincides with $\mathsf{F}^{B,N}_{ks}$ when $B = N = q$.

2.5 Our Realization of FE for Keyword Search

We realize our FE scheme for the keyword search functionality $\mathsf{F}_{ks}^{B,N}$ through a more powerful scheme for the so-called *orthogonality* functionality [KSW08]. In the latter we have the message space $\mathcal{X} = \mathbb{F}_q^n$ and each function $f_{\mathbf{y}}$ is defined by a vector $\mathbf{y} \in \mathbb{F}_q^n$ and is such that $f_{\mathbf{y}}(\mathbf{x})$ returns 1 when $\mathbf{y}^\top \mathbf{x} = 0$, and 0 otherwise.

A construction of FE for F_{ks} from FE for orthogonality already appears in previous work [KSW08]. In this paper, we tweak this construction in order to support the more general functionality $\mathsf{F}_{ks}^{B,N}$ described earlier. We present our construction in Fig. 1. The idea is to use the FE for orthogonality (with dimension $n = 2$) so that: encryption of a value $x \in [B]$ is an encryption of the vector $\mathbf{x}' = (x, -1)$, and a key for $y \in [N]$ is a collection of keys for the vectors $\mathbf{y}_i = (1, y + i \cdot N)$, with $i \in [k]$ where k is such that $B \leq k \cdot N$. This way, decryption can be realized by testing if one of the keys successfully decrypts, which is justified by the fact that, with these encodings, $x = y$ holds if $\mathbf{y}_i^\top \mathbf{x}' = 0$ for an $i \in [k]$.

```
$\mathsf{KS.Setup}(1^\lambda, B, N)$:
$(\mathsf{mpk}', \mathsf{msk}') \xleftarrow{\$} \mathsf{FE.Setup}(1^\lambda, 2)$
Let $k$ be the smallest integer s.t. $B \leq kN$
return $\mathsf{mpk} = (\mathsf{mpk}', B, N, k), \mathsf{msk} = \mathsf{msk}'$
```

```
$\mathsf{KS.Enc}(x, \mathsf{mpk})$:
return $c \xleftarrow{\$} \mathsf{FE.Enc}((x, -1), \mathsf{mpk})$
```

```
$\mathsf{KS.KeyGen}(y, \mathsf{msk})$:
for $i = 0 \ldots k-1$ do
    $\mathsf{sk}_i \xleftarrow{\$} \mathsf{FE.KeyGen}((1, y + i \cdot N), \mathsf{msk})$
return $\mathsf{sk}_y \leftarrow (\mathsf{sk}_0, \ldots, \mathsf{sk}_{k-1})$
```

```
$\mathsf{KS.Dec}(c, y, \mathsf{mpk}, \mathsf{sk}_y)$:
if $\exists i \in [k] : \mathsf{FE.Dec}(c, y + iN, \mathsf{mpk}, \mathsf{sk}_i) = 1$
    return 1
else return 0
```

Fig. 1: Our FE for $\mathsf{F}_{ks}^{B,N}$ from FE for orthogonality

This scheme is secure under a weaker notion in which the adversary asks secret keys for keywords y such that $y \neq x_0 \mod N$ and $y \neq x_1 \mod N$. This restriction (which corresponds to the notion of weak attribute-hiding) is sufficient in our application as we want to hide the winning index x only from those users that will not win i.e. from those holding keys for $y \neq x \mod N$.

2.6 Our Construction of FE for orthogonality

Among the known constructions of FE for orthogonality, we choose the pairing-based one proposed in [Wee17] for its MPC-friendliness. However, as the construction of [Wee17] achieves a stronger security definition for a larger functionality and under weaker assumptions than SXDH, we tailor and simplify their solution to meet our goals in the most efficient way. Our variant is detailed in Figure 2 while in the appendix, Section D.1 we prove the following theorem.

Proposition 1. *The scheme in Fig. 2 is selective secure under the* SXDH *assumption*

```
$\mathsf{FE.Setup}(1^\lambda, n)$:
    $\mathbf{a}, \mathbf{u}, \mathbf{w}_1, \dots, \mathbf{w}_n \leftarrow^{\$} \mathbb{F}_q^2$ s. t. $\mathbf{a}^\top \mathbf{u} \neq 0$
    $\mathsf{mpk} \leftarrow \left([\mathbf{a}]_1, [\mathbf{a}^\top \mathbf{u}]_1, [\mathbf{a}^\top \mathbf{w}_1]_1, \dots, [\mathbf{a}^\top \mathbf{w}_n]_1\right)$
    $\mathsf{msk} \leftarrow (\mathbf{w}_i)_{i=1}^n$ and **return** $(\mathsf{mpk}, \mathsf{msk})$

$\mathsf{FE.KeyGen}(\mathbf{y}, \mathsf{msk})$:
    $r \leftarrow^{\$} \mathbb{F}_q \setminus \{0\}$
    **return** $\mathsf{sk}_\mathbf{y} \leftarrow \left[\sum_{i=1}^n r y_i \mathbf{w}_i\right]_2, [r]_2$

$\mathsf{FE.Dec}(c, \mathbf{y}, \mathsf{mpk}, \mathsf{sk}_\mathbf{y})$:
    Parse $c = (\mathbf{c}_0, c_1, \dots, c_n)$ with $\mathbf{c}_0 \in \mathbb{G}_1^2$
    Parse $\mathsf{sk}_\mathbf{y} = (\mathbf{d}_0, d_1)$ with $\mathbf{d}_0 \in \mathbb{G}_2^2$
    **if** $d_1 \neq g_2^0$ **return** $e(\mathbf{c}_0, \mathbf{d}_0) \overset{?}{=} e(c_1^{y_1} \cdots c_n^{y_n}, d_1)$

$\mathsf{FE.Enc}(\mathbf{x}, \mathsf{mpk})$:
    $s \leftarrow^{\$} \mathbb{F}_q \setminus \{0\}$
    $c_i \leftarrow \left[s\mathbf{a}^\top (x_i \mathbf{u} + \mathbf{w}_i)\right]_1$
    **return** $c \leftarrow \left([s\mathbf{a}]_1, c_1, \dots, c_n\right)$
```

Fig. 2: Our simplified version of [Wee17] FE scheme for orthogonality

2.7 Non Interactive Zero-Knowledge

A non-interactive zero-knowledge (NIZK) proof system for a relation $\mathcal{R}$ is a tuple of PPT algorithms $(\mathsf{NIZK.G}, \mathsf{NIZK.P}, \mathsf{NIZK.V})$ where: $\mathsf{NIZK.G}$ on input the security parameter outputs a common reference string crs; $\mathsf{NIZK.P}(\mathsf{crs}, x, w)$, given $(x, w) \in \mathcal{R}$, outputs a proof π; $\mathsf{NIZK.V}(\mathsf{crs}, x, \pi)$, given statement x and proof π outputs 0 (reject) or 1 (accept). We say that a NIZK for $\mathcal{R}$ is *correct* if for every $\mathsf{crs} \leftarrow^{\$} \mathsf{NIZK.G}(1^\lambda)$ and all $(x, w) \in \mathcal{R}$, $\mathsf{NIZK.V}(\mathsf{crs}, x, \mathsf{NIZK.P}(\mathsf{crs}, x, w)) = 1$ holds with probability 1. Note that for ease of notation we usually omit the crs from the input. In our protocols we require the NIZKs to satisfy the notions of weak simulation extractability [Sah99] and zero-knowledge [FLS90].
About the first property, we remark that it only guarantees the extractability of proofs produced by the adversary that are not equal to proofs previously observed. For this reason we make them "unique" by adding implicitly a session ID to the statement. Concretely this means that in the Fiat Shamir transform, the hash function evaluation needs to be salted with the unique session ID. Finally we won't detail how to handle these *sid*, in the same way we don't detail it for ideal functionalities invocations.

We now define three relations useful when dealing with group elements. The first one checks whether two vectors $\mathbf{g}, \mathbf{h} \in \mathbb{G}_1$ are proportional, i.e., there exists $a \in \mathbb{F}_q$ s.t. $\mathbf{g}^a = \mathbf{h}$. The second one is a generalisation of the former and asks for the existence of solutions to the system $AX = B$ for A, B matrices given in the exponent. Finally the last one checks that an ElGamal ciphertext encrypts a message lying in a given range.

$$\begin{aligned}
\mathcal{R}_{\mathsf{DDH}} &= \{((\mathbf{g}, \mathbf{h}), s) \ : \ \mathbf{g}, \mathbf{h} \in \mathbb{G}^n, \ \mathbf{g}^s = \mathbf{h}\} \\
\mathcal{R}_{\mathsf{Lin}} &= \left\{(([A]_1, [B]_1), X) \ : \ A \in \mathbb{F}_q^{k,m}, \ B \in \mathbb{F}_q^{k,n}, \ X \in \mathbb{F}_q^{m,n}, \ AX = B\right\} \\
\mathcal{R}_{\mathsf{Rng}} &= \{((g, h, k, g^r, h^r k^m, [B]), (r, m)) \ : \ g, h, k \in \mathbb{G}_1, \ r \in \mathbb{F}_q, \ m \in [B] \subseteq \mathbb{F}_q\}
\end{aligned}$$

Moreover, as players need to prove their ability to decrypt a given ciphertext, we introduce $\mathcal{R}_{\mathsf{Dec}}$ a relation based on the language $\mathcal{L}_{\mathsf{key}}$ that captures all the secret keys satisfying correctness. We remark that this relation does *not* implies that sk was correctly generated.

$$\begin{aligned}
\mathcal{L}_{\mathsf{key}} = \{(\mathsf{mpk}, f, \mathsf{sk}) \ : \ \forall x, r \quad c = \mathsf{FE.Enc}(x, \mathsf{mpk}; r) \ \Rightarrow \\
\Rightarrow \ \mathsf{FE.Dec}(c, f, \mathsf{mpk}, \mathsf{sk}) = f(x)\}
\end{aligned}$$

$$\mathcal{R}_{\mathsf{Dec}} = \{((\mathsf{mpk}, c, f), \mathsf{sk}) \ : \ (\mathsf{mpk}, f, \mathsf{sk}) \in \mathcal{L}_{\mathsf{key}}, \ \mathsf{FE.Dec}(c, f, \mathsf{mpk}, \mathsf{sk}) = 1\}$$

To construct our protocols, we assume the existence of a NIZK argument for each of these relations. We note that there exists a sigma protocol for each of these relations, and that Fiat-Shamir based NIZKs from sigma protocols can be proven weakly-simulation-extractable [FKMV12] based on a special property of sigma protocols called quasi-unique responses (which is essentially satisfied by all Schnorr-like protocols where the third message is uniquely determined given the previous two). For the relations $\mathcal{R}_{\mathsf{DDH}}$ and $\mathcal{R}_{\mathsf{Lin}}$, we can use generalised Schnorr sigma protocols provided in [Mau15]. For the proof of range,[6] we propose a variant of the folklore solution based on binary decomposition, detailed in the appendix, Section B.2. We propose a sigma protocol for $\mathcal{R}_{\mathsf{Dec}}$ in appendix B.1.

2.8 UC model and Ideal Functionalities

The celebrated UC model, introduced in the seminal work of Ran Canetti [Can01], is a framework that allows to prove security properties of a protocol that are preserved under composition. This is done by comparing the protocol to an ideal functionality $\mathcal{F}$ defined to capture the intended properties. A protocol securely realise $\mathcal{F}$ if there exists a simulator $\mathcal{S}$ such that the composition $\mathcal{F} \circ \mathcal{S}$ is indistinguishable from the real protocol. The distinguisher $\mathcal{Z}$, also called the *environment*, is granted the power to choose all parties input, learn their output and corrupt any number of parties learning their internal state and influencing their behaviour. The challenge for $\mathcal{S}$ is therefore to reproduce all the messages sent by uncorrupted parties in a consistent way with their input/output, even though $\mathcal{S}$ cannot access it. To make this possible in non trivial cases, functionalities are often designed to leak some information to $\mathcal{S}$ and allow the simulator to influence the result in some way. Below we define three functionalities required in our construction: $\mathcal{F}_{\mathsf{Com}}$ to model commitments, $\mathcal{F}_{\mathsf{Czk}}$ to model committed NIZK Proof of Knowledge and $\mathcal{F}_{\mathsf{CT}}^{\mathcal{D}}$ to model a random beacon.
The first one was introduced in [CF01]. Our definition slightly differ as all the recipients receive the same value in the opening phase. This deviation is justified by our assumption of an authenticated broadcast channel. The second one resemble the $\mathcal{F}_{\mathsf{zk}}$ functionality introduced in [CF01] but the interaction is divided in a commitment phase and in proof phase. In practice $\mathcal{F}_{\mathsf{Czk}}$ can be efficiently realised sending a commitment to an *online-extractable* NIZK discussed in [Fis05]. The latter was recently introduced in [CD20] and realised assuming a suboptimal honest majority under standard assumptions.
All these functionalities can be realised more efficiently assuming a global random oracle in the GUC model [CDPW07]. For instance in [CDG+18] the security of the folklore commitment $H(m||r)$ is proved assuming a restricted programmable and observable random oracle.

2.9 Single Secret Leader Election

In this section we present the notion of a *single secret leader election* scheme as formalized by Boneh et al. [BEHG20] using a game-based approach. Informally, Boneh et al. define SSLE as a collection of protocols that allow a set of users to setup the system, register to be eligible or quit, secretly elect a single leader among registered users, claim victory and verify these claims. Security is captured by three (game-base) properties *uniqueness*, *fairness* and *unpredictability* respectively implying that

[6] For this, the most efficient choice to date may be an adaptation of Bulletproofs [BBB+18]; however, to the best of our knowledge, this is not known to be simulation-sound.

The Commitment Functionality $\mathcal{F}_{\mathsf{Com}}$:

– Upon receiving $(\mathsf{commit}, sid, m)$ from P_i, if there is no record of $(sid, i, \cdot)$ store (sid, i, m) and broadcast $(\mathsf{recepit}, sid, i)$.

– Upon receiving (open, sid) from P_i if (sid, i, m) was previously stored broadcast $(\mathsf{decom}, sid, i, m)$. Ignore further request on sid from P_i.

The Committed ZK Functionality $\mathcal{F}^{\mathcal{R}}_{\mathsf{Czk}}$:

– Upon receiving $(\mathsf{prove}, sid, x, w)$ from P_i, if there is no record of $(sid, i, \cdot)$ and $(x, w) \in \mathcal{R}$ store (sid, i, x) and broadcast $(\mathsf{recepit}, sid, i)$.

– Upon receiving (open, sid) from P_i if (sid, i, x) was previously stored broadcast $(\mathsf{proof}, sid, i, x)$. Ignore further request on sid from P_i.

The Coin Tossing Functionality $\mathcal{F}^{\mathcal{D}}_{\mathsf{CT}}$:
Parametrized by a distribution $\mathcal{D}$. Upon receiving (toss, sid) from all the honest parties, sample $x \xleftarrow{\$} \mathcal{D}$ and broadcast $(\mathsf{tossed}, sid, x)$

there is only one elected leader, that the election is not biased, and that the adversary has no information on the winner until she reveals herself.

In the definitions given below we slightly depart from the syntax and games in [BEHG20]. First, [BEHG20] assumes a protocol to verify the registration of a given user: we incorporate this step as part of the registration protocol. Second, we add a revocation protocol that removes a player from future elections: this change was already suggested by [BEHG20], but not fully formalized. The third and more relevant difference is that, in contrast to the definitions in [BEHG20], our security games further require that no sub protocol aborts, i.e. halts without returning any input. To justify this addition we observe that without it, security is not guaranteed for protocols that restart a sub procedure that has failed. To the best of our understanding, this assumption is de facto present in the security proofs of all schemes in [BEHG20]. The fourth difference is that we consider two threshold parameters to model the maximum number of users that the adversary can corrupt: t is a threshold over the *total* number N of users in the system (i.e., the adversary can corrupt up to t out of N users); $\vartheta(n)$ is instead a threshold function over the number of *registered* users in an election (i.e., the adversary can corrupt $< \vartheta(n)$ users that are registered in the election). This change is done to let the definition capture more constructions: indeed there may be schemes where one needs honest majority over the total number of users (e.g., because they all hold a share of a global secret key), and others that can be secure even when the adversary controls a majority of all the users, but security of a given election still needs an honest majority of the users that participate in it.

Finally, we recall that our constructions satisfy a UC-secure notion of SSLE that we introduce in the next section. We provide the following game-based notion for the sake of a comparison. As shown in the next section, game-based SSLE is implied by UC-secure SSLE, while the converse may not hold.

Definition 5. *A **Secret Single Leader Election** is a tuple of six protocols* $(\mathsf{SSLE.Setup}, \mathsf{SSLE.Reg}, \mathsf{SSLE.Rev}, \mathsf{SSLE.Elect}, \mathsf{SSLE.Claim}, \mathsf{SSLE.Vrf})$ *executed among N users, such that*

- $\mathsf{SSLE.Setup}$ *returns* pp *to every player and* sp_i *to* P_i.
- $\mathsf{SSLE.Reg}_{\mathsf{pp}}(i)$ *registers player P_i for future elections*
- $\mathsf{SSLE.Rev}_{\mathsf{pp}}(i)$ *revokes the registration of player P_i from 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.*

Definition 6 (Uniqueness). *An SSLE scheme satisfies (t, ϑ)-**threshold uniqueness** if for all* $\mathsf{PPT}\mathcal{A}$ *that corrupt $T < t$ parties there exists a negligible function ε s.t.*

$$\Pr\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{Uniq}}(1^\lambda, N, \vartheta) = 1\right] \leq \varepsilon(\lambda).$$

*If $t = N$ and $\vartheta = 1_{\mathbb{N}}$ (i.e. ϑ is the identity function) we simply say that the scheme satisfies **uniqueness**.*

Definition 7 (Fairness). *An SSLE scheme satisfies (t, ϑ)-**threshold** $\eta(\kappa)$-**fairness** if for every* PPT*algorithm $\mathcal{A}$ that corrupts $T < t$ players there exists a negligible function ε such that*

$$\left|\Pr\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{Fair}}(1^\lambda, N, \vartheta) = 1\right] - \frac{n-\tau}{n}\right| \leq \varepsilon(\lambda) + \eta(\kappa).$$

*where n and τ are defined in the experiment. If $\eta(\kappa) = 0$ we say that the scheme satisfies (t, ϑ)-**threshold fairness**. Finally if $T = N$ and $\vartheta = 1_{\mathbb{N}}$ we simply say that scheme satisfies **fairness**.*

Definition 8 (Unpredictability). *An SSLE scheme satisfies (t, ϑ)-**threshold** $\eta(\kappa)$-**unpredictability** if for every* $\mathsf{PPT}\mathcal{A}$ *that corrupts $T < t$ parties there exists a negligible function ε such that*

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

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.*
*If $\eta(\kappa) = 0$ the scheme satisfies (t, ϑ)-**threshold unpredictability**. If $t = N$ and $\vartheta = 1_{\mathbb{N}}$ the scheme is simply said to satisfy **unpredictability**.*

3 Universally Composable SSLE

The game-based security definitions given in the previous section (i.e., definitions 6, 7, and 8) capture the three essential properties an SSLE scheme should have. Yet, the security experiments do not model scenarios where multiple executions of the setup/registration/election protocols may occur concurrently. Moreover, as in most game-based notion, security is not guaranteed to hold when the primitive is used in a more complex protocol.

For this reason, we propose a definition of SSLE in the universal composability model. To this end, we define an ideal functionality $\mathcal{F}_{\mathsf{SSLE}}$ that performs elections and reveals the winners in an ideal way. A UC-secure SSLE scheme is then any protocol that securely realizes $\mathcal{F}_{\mathsf{SSLE}}$.

Uniqueness Experiment $\mathsf{Exp}^{\mathcal{A}}_{\mathsf{Uniq}}(1^\lambda, N, t, \vartheta)$:

```
1 : When $M \leftarrow \mathcal{A}(1^\lambda, N)$, simulate $P_i$ for $i \in [N] \setminus M$ interacting with $\mathcal{A}$
2 : Execute $\mathsf{SSLE.Setup} \rightarrow \mathsf{pp}, \mathsf{sp}_i$ 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 $\mathsf{revoke}, i \leftarrow \mathcal{A}$: Run $\mathsf{SSLE.Rev}_{\mathsf{pp}}(i)$ and remove $R \leftarrow R \setminus \{i\}$
5 : When $\mathsf{elect} \leftarrow \mathcal{A}$ and $|R \cap M| < \vartheta(|R|)$: Execute $\mathsf{SSLE.Elect}_{\mathsf{pp}} \rightarrow c_s$
6 :   $\pi_{i,s} \leftarrow \mathsf{SSLE.Claim}_{\mathsf{pp}}(c_s, \mathsf{sp}_i, i)$,   $\mathcal{A} \leftarrow \pi_{i,s}$   $\forall i \in R \setminus M$
7 :   $\pi_{i,s} \leftarrow \mathcal{A}$   $\forall i \in R \cap M$;   $s \leftarrow s + 1$
8 : When $\mathcal{A}$ halts, return 1 if any protocol failed or if $\exists s' \in [s]$ and $i, j \in [N]$ distinct such
    that $\mathsf{SSLE.Vrf}_{\mathsf{pp}}(c_t, \pi_{i,s'}, i) = \mathsf{SSLE.Vrf}_{\mathsf{pp}}(c_t, \pi_{j,s'}, j) = 1$
```

Fairness Experiment $\mathsf{Exp}^{\mathcal{A}}_{\mathsf{Fair}}(1^\lambda, N, t, \vartheta)$:

```
1 : When $M \leftarrow \mathcal{A}(1^\lambda, N)$, simulate $P_i$ for $i \in [N] \setminus M$ interacting with $\mathcal{A}$
2 : Execute $\mathsf{SSLE.Setup} \rightarrow \mathsf{pp}, \mathsf{sp}_i$ 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 $\mathsf{revoke}, i \leftarrow \mathcal{A}$: Run $\mathsf{SSLE.Rev}_{\mathsf{pp}}(i)$ and remove $R \leftarrow R \setminus \{i\}$
5 : When $\mathsf{elect} \leftarrow \mathcal{A}$ and $|R \cap M| < \vartheta(|R|)$: Execute $\mathsf{SSLE.Elect}_{\mathsf{pp}} \rightarrow c_s$
6 :   $\pi_{i,s} \leftarrow \mathsf{SSLE.Claim}_{\mathsf{pp}}(c_s, \mathsf{sp}_i, i)$,   $\mathcal{A} \leftarrow \pi_{i,s}$   $\forall i \in R \setminus M$
7 :   $\pi_{i,s} \leftarrow \mathcal{A}$   $\forall i \in R \cap M$;   $s \leftarrow s + 1$
8 : When $\mathsf{chall} \leftarrow \mathcal{A}$ and $|R \cap M| < \vartheta(|R|)$: Call $n = |R|$ and $\tau = |R \cap M|$;
9 :   Execute $\mathsf{SSLE.Elect}_{\mathsf{pp}} \rightarrow c_s$
10 :   $\pi_{i,s} \leftarrow \mathsf{SSLE.Claim}_{\mathsf{pp}}(c_s, \mathsf{sp}_i, i)$   $\forall i \in R \setminus M$
11 :   Return 1 if no protocol fails and $\exists i \in R \setminus M$ such that $\mathsf{SSLE.Vrf}_{\mathsf{pp}}(c_s, \pi_{i,s}, i) = 1$, 0
     otherwise
```

At a high-level, $\mathcal{F}_{\mathsf{SSLE}}$ consists of the following commands. By using ($\mathsf{register}$) and (revoke), a user can register to and be removed from an election. When all the honest users call (elect, eid), a new election with identifier eid is performed, that is the ideal functionality samples a winner index j uniformly at random from the set of registered users. By using the (elect, eid) command, every honest user is informed by the ideal functionality on whether she is the winner of the election eid. Using (reveal, eid), an honest winning user instructs the ideal functionality to announce the election's outcome to everyone. Finally, the ($\mathsf{fake\_rejected}, eid, j$) command is reserved to the adversary to instruct the ideal functionality to announce to everyone that the (corrupted) user j is *not* the winner. This model a scenario in which an adversary deviates from the protocol to claim victory in spite of being the leader. The formal definition of the $\mathcal{F}_{\mathsf{SSLE}}$ functionality is given below.

In order to emulate the (t, ϑ)-threshold definitions in Section 2.9 we need to specify a class of environments that (1) corrupts M with $|M| < t$ parties; (2) induce $\mathcal{F}_{\mathsf{SSLE}}$ to perform elections only when $|R \cap M| < \vartheta(|R|)$. Those properties are formally captured by the following definition

Definition 9. *A protocol Π is said to (t, ϑ)-threshold realise $\mathcal{F}_{\mathsf{SSLE}}$ if there exists a simulator $\mathcal{S}$ such that Π is indistinguishable from $\mathcal{F}_{\mathsf{SSLE}} \circ \mathcal{S}$ for all* PPT *environments $\mathcal{Z}$ that statically corrupt a set M of parties with $|M| < t$ and such that, each time an honest player returns* (won, eid, R) *or* (lost, eid, R) *then $|R \cap M| < \vartheta(|R|)$.*

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 $\mathsf{SSLE.Setup} \rightarrow \mathsf{pp}, \mathsf{sp}_i$ 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 $\mathsf{revoke}, i \leftarrow \mathcal{A}$: Run $\mathsf{SSLE.Reg}_{\mathsf{pp}}(i)$ and remove $R \leftarrow R \setminus \{i\}$
5 :  When $\mathsf{elect} \leftarrow \mathcal{A}$ and $|R \cap M| < \vartheta(|R|)$: Execute $\mathsf{SSLE.Elect}_{\mathsf{pp}} \rightarrow c_s$
6 :    $\pi_{i,s} \leftarrow \mathsf{SSLE.Claim}_{\mathsf{pp}}(c_s, \mathsf{sp}_i, i), \quad \mathcal{A} \leftarrow \pi_{i,s} \quad \forall i \in R \setminus M$
7 :    $\pi_{i,s} \leftarrow \mathcal{A} \quad \forall i \in R \cap M; \quad s \leftarrow s + 1$
8 :  When $\mathsf{chall} \leftarrow \mathcal{A}$ and $|R \cap M| < \vartheta(|R|)$: Call $n = |R|$ and $\tau = |R \cap M|$;
9 :    Execute $\mathsf{SSLE.Elect}_{\mathsf{pp}} \rightarrow c_s$
10 :   $\pi_{i,s} \leftarrow \mathsf{SSLE.Claim}_{\mathsf{pp}}(c, \mathsf{sp}_i, i) \quad \forall i \in R \setminus M; \quad j \leftarrow \mathcal{A}$
11 :   Return 1 if any protocol fails or if $\mathsf{SSLE.Vrf}_{\mathsf{pp}}(c_s, \pi_{j,s}, j) = 1$, 0 otherwise
```

The SSLE functionality $\mathcal{F}_{\mathsf{SSLE}}$:
Initialise $E, R \leftarrow \varnothing$ and let M be the set of corrupted parties. Upon receiving:

- $(\mathsf{register})$ from P_i: if $i \notin R$, add $R \leftarrow R \cup \{i\}$ and broadcast $(\mathsf{registered}, i)$.
- (revoke) from P_i: if $i \in R$, remove $R \leftarrow R \setminus \{i\}$ and broadcast $(\mathsf{revoked}, i)$.
- (elect, eid) from all honest parties: if $R \neq \varnothing$ and eid was not requested before sample $j \xleftarrow{\$} R$ and send (won, eid, R) to P_j and (lost, eid, R) to P_i for $i \in R \setminus \{j\}$. Store $E \leftarrow E \cup \{(eid, j, R)\}$.
- (reveal, eid) from P_i: if $(eid, j, R') \in E$ and $i \in R'$, broadcast $(\mathsf{result}, eid, R', i)$ if $i = j$, $(\mathsf{rejected}, eid, R', i)$.
- $(\mathsf{fake\_rejected}, eid, j)$ from the adversary: if $(eid, \cdot, R') \in E$, $j \in R'$, and P_j is corrupted broadcast $(\mathsf{rejected}, eid, R', j)$.

Definition 10. *A (t, ϑ)-**threshold statically secure UC-SSLE** is a protocol Π that (t, ϑ)-securely realise $\mathcal{F}_{\mathsf{SSLE}}$. If $t = N$ and $\vartheta = 1_{\mathbb{N}}$ then Π is called a **statically secure UC-SSLE**.*

To further motivate our UC-secure notion of SSLE we compare it to the game-based notion of the previous section. First, in the following Proposition, we show that the UC notion implies the game-based one. A proof appears in the appendix, Section C.1

Proposition 2. *If Π is a (t, ϑ)-threshold statically secure UC-SSLE protocol, then its derived SSLE scheme described in Figure 3 satisfies (t, ϑ)-threshold uniqueness, (t, ϑ)-threshold fairness and (t, ϑ)-threshold unpredictability.*

Second, we argue that our UC notion is strictly stronger than the game-based one. For this, we simply observe that we can take one of the protocols from [BEHG20] (e.g., the one based on TFHE or the one based on Shuffling) and notice that they cannot be UC-secure if the zero-knowledge

```
SSLE.Setup:
    $P_i$ sets $(\mathsf{pp}, \mathsf{sp}_i, \mathsf{cnt}) \leftarrow (\perp, \perp, 0)$

SSLE.Elect_pp:
    All players send $(\mathsf{elect}, \mathsf{cnt})$ to $\Pi$
    $\mathsf{cnt} \leftarrow \mathsf{cnt} + 1$
    When $(\mathsf{won}, eid, R) \leftarrow \Pi$ or
    When $(\mathsf{lost}, eid, R) \leftarrow \Pi$:
        Return $c \leftarrow (eid, R)$

SSLE.Reg_pp(i):
    $P_i$ sends $(\mathsf{register}, i)$ to $\Pi$
    Others wait $(\mathsf{registered}, i) \leftarrow \Pi$

SSLE.Claim_pp(c, sp_i, i):
    Send $(\mathsf{reveal}, c)$. Return $\pi \leftarrow \perp$

SSLE.Rev_pp(i):
    $P_i$ sends $(\mathsf{revoke}, i)$ to $\Pi$
    Others wait $(\mathsf{revoked}, i) \leftarrow \Pi$

SSLE.Vrf_pp(c, π, i):
    When $(\mathsf{result}, c, i) \leftarrow \Pi$ return 1
    When $(\mathsf{rejected}, c, i) \leftarrow \Pi$ return 0
```

Fig. 3: The derived SSLE scheme from a UC-SSLE protocol Π

proofs they employ are not UC-secure.[7] In [BEHG20], these protocols are proven secure without making any UC assumption on these zero-knowledge proofs; so they constitute a counterexample of protocols that are secure in the game-based sense but would not be secure according to our UC notion.

3.1 A weaker definition

Definition 10 provides a higher level of security with respect to the game-based definition, but at the same time requires more structure from the underlying protocol and therefore may imply higher costs. For this reason we chose to present here a weaker functionality $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ that is static, i.e. does not accept register or revoke requests and allows the adversary to control a certain election with probability smaller than $2^{-\kappa}$.

The reason we add dummy parties to the definition is technical as it allows external players to check the result of an election without actively taking part. Thanks to this addition in Section C.3, we show how to realise $\mathcal{F}_{\mathsf{SSLE}}$ in the $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$-hybrid model when $\kappa = \Theta(\lambda)$. The idea is to emulate register and verify requests by invoking a new $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ functionality with different *sid*.

At the same time providing more freedom on the choice of κ formalizes the intuition that in practice weaker forms of fairness and unpredictability might be sufficient, especially if these lead to significant efficiency gains. In Section C.2 we show how to construct a SSLE scheme (from a generic protocol realizing $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ and tolerating less than $\vartheta(n)$ corrupted players) that achieves $2^{-\kappa}$-fairness and $\eta(\kappa)$-unpredictability with

$$\eta(\kappa) \;=\; \max_{n \in [N]+1} \left(\frac{n}{n - \vartheta(n)} \right) \frac{1}{2^{\kappa}}.$$

For fairness, the $2^{-\kappa}$ bound simply means that for $k = \log N$ an adversary controlling T parties, wins the election with probability less than $(T+1)/N$. This is the same winning probability of an

[7] Here, as the candidate protocol we are assuming the one where each sub protocol is used to implement the corresponding command, i.e., SSLE.Reg for register, SSLE.Elect for elect, etc.

The Static Parametrised SSLE functionality $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$:
Executed with parties $P_1, \dots, P_N$, dummy parties $\bar{P}_1, \dots, \bar{P}_{\bar{N}}$ and malicious users $M \subseteq [N]$. Initially set $E \leftarrow \varnothing$. Upon receiving:

- (elect, eid) from all honest parties: if eid was not requested before leak $(\mathsf{electing}, eid)$. Upon receiving (prob, eid, p) with $p \leq 2^{-\kappa}$:
  With probability p instead leak $(\mathsf{corrupted}, eid)$ and wait for the adversary to reply with (infl, eid, j). Otherwise sample $j \leftarrow^{\$} [N]$.
  Send (won, eid) to P_j and (lost, eid) to P_i for $i \in [N] \setminus \{j\}$. Add $E \leftarrow E \cup \{(eid, j)\}$.

- (reveal, eid) from P_i with $(eid, j) \in E$: broadcast $(\mathsf{result}, eid, i)$ if $j = i$, otherwise broadcast $(\mathsf{rejected}, eid, i)$

- $(\mathsf{fake\_rejected}, eid, j)$ from the adversary: if $(eid, \cdot) \in E$ and P_j is corrupted broadcast $(\mathsf{rejected}, eid, j)$

adversary that runs a fair election and corrupts one single extra player! More precise details are in appendix C.3

4 UC-secure SSLE from FE for Keyword Search

In this section we present a generic construction of a UC-SSLE protocol based on functional encryption for keyword search, which, besides being of independent interest, serves as a stepping stone toward our full construction. More specifically we realise $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ assuming the existence of a protocol Π that securely performs a distributed key generation and on request produces ciphertexts encrypting messages uniformly distributed.
The protocol works as follows: Initially the keys $\mathsf{mpk}, \mathsf{sk}_i$ are distributed among N users with sk_i being associated to the keyword i. Each time an election is requested, users invokes Π which generate a challenge ciphertext c encrypting a message $m \in [N]$ and they check whether they won or lost by decrypting. Whoever can decrypt c to 1 is the leader and can claim victory by broadcasting a NIZK argument of this. Finally other users accept the claim if the NIZK argument given is correct.
To proceed we formally define a functionality $\mathcal{F}_{\mathsf{SnC}}$, that shapes behaviour and security of Π, and a protocol $\{P^{(i)}_{\mathsf{KS-SSLE}} : i \in [N]\}$ in the $\mathcal{F}_{\mathsf{SnC}}$-hybrid model that realises $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$. The following theorem is proven in Appendix D.2.

Theorem 1 *The protocol $\{P^{(i)}_{\mathsf{KS-SSLE}} : i \in [N]\}$ securely realises $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$, for any κ, in the $\mathcal{F}_{\mathsf{SnC}}$-hybrid model for the class of* PPT *environments $\mathcal{Z}$ that statically corrupts up to N players.*

5 An Efficient UC-secure SSLE from SXDH

In this section we propose our main contribution, an SSLE protocol that works over bilinear groups and that we prove UC-secure under the SXDH assumption.

The Setup and Challenge Functionality $\mathcal{F}_{\mathsf{SnC}}$:

Generate $\mathsf{crs} \leftarrow^{\$} \mathsf{NIZK.G_{Dec}}(1^\lambda)$, $\mathsf{mpk}, \mathsf{msk} \leftarrow^{\$} \mathsf{KS.Setup}(1^\lambda)$ and $\mathsf{sk}_i \leftarrow^{\$} \mathsf{KS.KeyGen}(i, \mathsf{msk})$ for all $i \in [N]$. Upon receiving

– (setup) from P_i: send $(\mathsf{input}, \mathsf{crs}, \mathsf{mpk}, \mathsf{sk}_i)$ to P_i

– $(\mathsf{ch\_request}, eid)$ from all honest parties: If eid was not requested before, sample $j \leftarrow^{\$} [N]$ and compute $c \leftarrow^{\$} \mathsf{KS.Enc}(j, \mathsf{mpk})$. Broadcast $(\mathsf{challenge}, eid, c)$

Party $P^{(i)}_{\mathsf{KS-SSLE}}$ realising $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$:

Initially set $C \leftarrow \varnothing$, send setup to $\mathcal{F}^{\kappa}_{\mathsf{SnC}}$ and wait for the reply $(\mathsf{input}, \mathsf{crs}, \mathsf{mpk}, \mathsf{sk}_i)$. Upon receiving input:

– (elect, eid): send $(\mathsf{ch\_request})$ to $\mathcal{F}^{\kappa}_{\mathsf{SnC}}$

– $(\mathsf{challenge}, eid, c)$ from $\mathcal{F}_{\mathsf{SnC}}$: store $C \leftarrow C \cup \{(eid, c)\}$. Return (won, eid) if $1 \leftarrow \mathsf{KS.Dec}(c, i, \mathsf{mpk}, \mathsf{sk}_i)$, otherwise return (lost, eid).

– (reveal, eid): if $(eid, c) \in C$ and $1 \leftarrow \mathsf{KS.Dec}(c, i, \mathsf{mpk}, \mathsf{sk}_i)$, prove $\pi \leftarrow^{\$} \mathsf{NIZK.P_{Dec}}(\mathsf{crs}, \mathsf{mpk}, c, i, \mathsf{sk}_i)$ and broadcast $(\mathsf{claim}, eid, \pi)$. Otherwise broadcast $(\mathsf{claim}, eid, \perp)$

– $(\mathsf{claim}, eid, \pi)$ from P_j: if $1 \leftarrow \mathsf{NIZK.V_{Dec}}(\mathsf{crs}, \mathsf{mpk}, c, i, \pi)$ return $(\mathsf{result}, eid, j)$ otherwise $(\mathsf{rejected}, eid, j)$

5.1 Intuition

The idea is to instantiate the generic construction of Section 4 with the keyword search FE scheme obtained applying the transformation in Figure 1 to our OFE in Figure 2.

The first challenge is to make the encryption algorithm MPC-friendly. To address this, we select a random committee $Q \subseteq [N]$ of players through a random beacon, that produces an encryption of an integer m obtained as the sum of $m_j \in [N]$ secretly chosen by P_j for $j \in Q$. A downside is that now $m \in [|Q|N]$. For this reason, we use the FE for the functionality $\mathsf{F}^{B,N}_{ks}$ with $B = |Q| \cdot N$. This way, decryption still provides a good way to test if one wins, as with this functionality the holder of secret key for index i learns if $m = i \bmod N$. Also, if at least one m_j is uniform over $[N]$, so is $m \bmod N$. Moreover, in our case B is still a small value as $|Q| \le \kappa$ and thus the decryption of our scheme of Fig. 1 is efficient.

Next step is to show in more detail how the committee can accomplish its task. The ciphertext we want to produce has the following shape:

$$\mathbf{c}_0 = [s\mathbf{a}]_1, \quad c_1 = \left[s\mathbf{a}^\top(m\mathbf{u} + \mathbf{w}_1)\right]_1, \quad c_2 = \left[s\mathbf{a}^\top(\mathbf{w}_2 - \mathbf{u})\right]_1.$$

$\mathbf{c}_0$ and c_2 are easy to generate in a distributed fashion: each party P_i can choose a random s_i, broadcast a commitment to $[s_i\mathbf{a}]_1$, $\left[s_i\mathbf{a}^\top(\mathbf{w}_2-\mathbf{u})\right]_1$ and then, after every party sent this, the commitments can be opened and each party can compute the product of all the received elements.

To produce c_1 we need more rounds of communication. The idea is to let every party send a commitment to $\left[m_i\mathbf{a}^\top\mathbf{u}\right]_1$ by using an homomorphic commitment[8] like ElGamal. This way, everyone can locally compute a commitment to

$$\left[s_i\mathbf{a}^\top\left(\sum\nolimits_{j\in Q} m_j\mathbf{u}-\mathbf{w}_1\right)\right]_1$$

through the homomorphic properties of the commitment (i.e., by taking the product of all the commitments, multiplying it by $\left[\mathbf{a}^\top\mathbf{w}_1\right]_1$, and finally exponentiating the result by s_i. Once these commitments are broadcasted and received, everyone can locally compute a commitment to $c_1 = \left[s\mathbf{a}^\top(m\mathbf{u}+\mathbf{w}_1)\right]_1$, where $s=\sum_i s_i$. In conclusion players jointly open c_1.

The reason our protocol is more elaborated than the idea presented so far is that, to the best of our knowledge, with all the homomorphic commitments the adversary can get the committed message *before* the honest parties in this opening phase,[9] and consequently it may refuse to open it if he figures out from the ciphertext that he lost the associated election. This issue would break the fairness property.

To solve this last issue we deploy a threshold ElGamal commitment. In this way, each time a player refuses to open its commitment, a threshold decryption can be performed in alternative. We remark that, even though reconstructions have linear communication complexity, each time one is performed a dishonest player is detected, upper bounding their number by t.

In order to achieve security against malicious players, our final protocol includes NIZK proofs to show consistency of the messages, e.g., that an m_i in the appropriate range $[N]$ is encrypted, or that the same s_i is used in the various group elements. Notably, all these NIZKs can be instantiated with very simple and efficient sigma protocols (see Section 2.7).

Finally, to complete the protocol we also show how to distribute the setup and key generation of our FE scheme. For ease of exposition, we first present a protocol assuming an ideal setup functionality in Section 5.2, and then in Section 5.3 we show how this functionality can be UC-realized.

5.2 SSLE protocol with Ideal Setup Functionality

We show a protocol that securely realizes the $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ ideal functionality for N players. To this end we use the following building blocks:

- The FE scheme for orthogonality in Fig. 2, denoted FE.
- NIZKs for relations $\mathcal{R}_{\mathsf{DDH}}, \mathcal{R}_{\mathsf{Rng}}$ and $\mathcal{R}_{\mathsf{Dec}}$. For readability, we suppress the crs from the inputs of the prover and verifier algorithm, e.g., $\mathsf{NIZK.P}_{\mathsf{DDH}}(\cdot,\cdot)$ stands for $\mathsf{NIZK.P}_{\mathsf{DDH}}(\mathsf{crs}_{\mathsf{DDH}},\cdot,\cdot)$.
- An ideal functionality $\mathcal{F}_{\mathsf{Com}}$ for commitments.
- An ideal functionality $\mathcal{F}_{\mathsf{Setup}}$ that generates the CRS of the above NIZKs and performs the $\mathsf{FE.Setup}$ algorithm and distributes to each player P_i a collection of secret keys for the vectors $\{(1, i+j\cdot N)\}_{j\in[\kappa]}$. The availability of this functionality is done to keep the exposition simple in this section.

[8] Supporting in this case the product of committed elements, the product of known elements and exponentiations

[9] One reason for this problem is that we are implicitly assuming a dishonest majority within the committee, where guaranteed output delivery cannot always be obtained.

– An ideal functionality $\mathcal{F}_{\mathsf{CT}}^{\kappa,D}$ for coin tossing that returns a random subset Q of $[N] \setminus D$ with cardinality ℓ, where ℓ is the smallest integer such that

$$\binom{t-|D|-1}{\ell}\binom{N-|D|}{\ell}^{-1} \leq 2^{-\kappa}.$$

Note that this implies that for any set M such that $D \subseteq M \subseteq [N]$ and $|M| < t$ then $\Pr[Q \subseteq M] \leq 2^{-\kappa}$. Also if $t \leq N/2$ then $\ell \leq \kappa$.

For ease of presentation, we let $\phi : [\kappa N] \to [N]$ be such that $\phi(m) = m \bmod N$. Its preimages are then sets of the form $\phi^{-1}(i) = \{(1, i + j \cdot N)\}_{j \in [\kappa]}$. The protocol is fully described in Fig. 4,5. Below we provide an explanation of its main steps.

The election protocol starts by having all parties calling the coin tossing functionality (line 1), which returns a randomly sampled committee $Q \subset [N] \setminus D$ (line 2). Next (lines 3–6), every player P_i in the selected committee chooses a random $m_i \in [N]$, creates an ElGamal commitment $(R_i, \widetilde{c}_{1,i})$ to $\left[m_i \mathbf{a}^\top \mathbf{u}\right]_1$ and implicitly commits in $(\widetilde{G}_i, \widetilde{H}_i)$ to a randomly sampled s_i, which represents its share of the randomness for the ciphertext to be created. P_i also creates two NIZKs π^i_{Rng} and $\pi^{0,i}_{\mathsf{DDH}}$ to prove, respectively, that m_i lies in the correct range $[N]$ and that $(\widetilde{G}_i, \widetilde{H}_i) = (g, h)^{s_i}$ (i.e., they share the same discrete log). All these messages are first broadcasted in committed form (line 6) and then, after all the players of the committee spoke or sent an error[10], opened (line 9). Whenever a committee member sends error, the honest participants exclude him from Q and, in case the committee set remained empty, a new committee is sampled by calling the coin tossing functionality.

Next (lines 11–15), upon receipt of the commitment opening, the players aggregate the answers and compute $G_i = g^{sr_i}$, $H_i = h^{sr_i}$, and a share of the ciphertext

$$\mathbf{c}_{0,i} \leftarrow [s_i \mathbf{a}]_1, \quad c_{1,i} = h^{rs_i}\left[s_i \mathbf{a}^\top(m\mathbf{u} + \mathbf{w}_1)\right]_1, \quad c_{2,i} \leftarrow \left[s_i \mathbf{a}^\top(\mathbf{w}_2 - \mathbf{u})\right]_1$$

along with three NIZK to prove the wellformedness of all these elements. All these values are broadcasted (line 17), except for H_i and the third NIZK which are instead broadcasted in committed form. If any error occurs parties remove detected player and repeat the previous step for the new set Q (lines 18–20).

This is the point up to which the protocol can be carried out in an offline phase. The online phase thus starts with every member of the committee executing line 21. As before users check the proof and compute c_1 removing the mask $H = \prod_{j \in Q} H_j$. Is some users, call A the set of their indices, refuse to send a correct H_j then honest users jointly generate $\prod_{j \in A} H_j$ from $\prod_{j \in A} G_j$ and their secret shares.

At the end (line 29), every player has the ciphertext $(\mathbf{c}_0, c_1, c_2)$ and can use his keys to check if he can decrypt it.

Finally, upon receiving (reveal, eid) the winner generates a NIZK to prove that it can decrypt the election's ciphertext. Every other player receiving a message $(\mathsf{claim}, eid, \pi, m)$ from P_j, can check the NIZK and if so to get convinced that P_j is the winner.

Theorem 2 *The protocol* $\{P^{(i)}_{\kappa,\mathsf{SSLE}} : i \in [N]\}$ *securely realizes* $\mathcal{F}^{\kappa}_{\mathsf{SSLE}}$ *in the* $(\mathcal{F}_{\mathsf{CT}}, \mathcal{F}_{\mathsf{Com}}, \mathcal{F}_{\mathsf{Setup}})$*-hybrid model under the* SXDH *assumption for the class of* PPT *environments* $\mathcal{Z}$ *that statically corrupts up* $\lfloor N/2 \rfloor$ *players*

[10] For simplicity, we assume that P_j sends error if no message is received from him before a timeout.

The Setup Functionality $\mathcal{F}_{\mathsf{Setup}}$
Generate $\mathsf{crs}_{\mathsf{DDH}} \xleftarrow{\$} \mathsf{NIZK}.\mathsf{G}_{\mathsf{DDH}}(1^\lambda), \mathsf{crs}_{\mathsf{Rng}} \xleftarrow{\$} \mathsf{NIZK}.\mathsf{G}_{\mathsf{Rng}}(1^\lambda)$,
$\mathsf{crs}_{\mathsf{Dec}} \xleftarrow{\$} \mathsf{NIZK}.\mathsf{G}_{\mathsf{Dec}}(1^\lambda)$
Sample $\mathsf{mpk}, \mathsf{msk} \xleftarrow{\$} \mathsf{FE}.\mathsf{Setup}(1^\lambda, 2)$, $g \xleftarrow{\$} \mathbb{G}_1$, $f \xleftarrow{\$} \mathbb{F}_q[x]_{\leq t}$.
For all $i \in [N]$ and $m \in \phi^{-1}(i)$ compute $\mathsf{sk}_{i,m} \xleftarrow{\$} \mathsf{FE}.\mathsf{KeyGen}((1, m), \mathsf{msk})$.
Set $\mathsf{sk}_i \leftarrow (\mathsf{sk}_{i,m})_{m \in \phi^{-1}(i)}$, $h \leftarrow g^{f(-1)}$ and $k_i = g^{f(i)}$.
Set $\mathsf{pp} \leftarrow (\mathsf{crs}_{\mathsf{DDH}}, \mathsf{crs}_{\mathsf{Rng}}, \mathsf{crs}_{\mathsf{Dec}}, \mathsf{mpk}, g, h, k_0, \ldots, k_{N-1})$, $\mathsf{sp}_i \leftarrow (\mathsf{sk}_i, f(i))$.

Upon receiving setup from P_i send $(\mathsf{input}, \mathsf{pp}, \mathsf{sp}_i)$.

5.3 Realising the Setup

For what regards $\mathsf{SSLE}.\mathsf{Setup}$, in Figure 6 we presented for simplicity a centralized version. Here we describe how to realise it in a distributed way.

First of all, in order to emulate private communication channels, not available in our model, that are necessary to distribute the secret parameters we assume that each user is associated to some ElGamal public key of which it knows the respective secret key. Second, we do not consider the realization of the NIZK crs generation as in our instantiations these are realized by applying Fiat-Shamir to sigma protocols and thus can be achieved by assuming a programmable random oracle ideal functionality. Next, in our realization we will not focus on the distribution of polynomial shares as this can be easily achieved through VSS modeled by the $\mathcal{F}_{\mathsf{VSS}}$ functionality. Our main focus will be on showing how to generate the public and secret keys of the FE scheme in Figure 2. To this aim, recall that

$$\mathsf{mpk} = [\mathbf{a}]_1\,, \left[\mathbf{a}^\top \mathbf{u}\right]_1, \left[\mathbf{a}^\top \mathbf{w}_1\right]_1, \left[\mathbf{a}^\top \mathbf{w}_2\right]_1, \quad \mathsf{sk}_{(1,m)} = [r\mathbf{w}_1 + rm\mathbf{w}_2]_2\,, [r]_2\,.$$

In the random oracle model $[\mathbf{a}]_1$ and $[r]_2$ can be uniformly sampled without any communication assuming two hash functions whose image is respectively $\mathbb{G}_1$ and $\mathbb{G}_2$. Moreover, when $\mathbf{a} \neq 0$, $\mathbf{a}^\top \mathbf{u}$ is uniform over $\mathbb{F}_q$ and independent from all the other components. Hence we can also generate $\left[\mathbf{a}^\top \mathbf{u}\right]_1$ through the random oracle.
For what regards the other components of $\mathsf{mpk}, \mathsf{sk}_{(1,m)}$, they now depend linearly only on $\mathbf{w}_1, \mathbf{w}_2$. Hence proceed choosing a random committee Q and letting every party P_i in Q sample $\mathbf{w}_{i,1}, \mathbf{w}_{i,2} \xleftarrow{\$} \mathbb{F}_q^2$ and compute $\left[\mathbf{a}^\top \mathbf{w}_{i,1}\right]_1$, $\left[\mathbf{a}^\top \mathbf{w}_{i,2}\right]_1$ and an encryption of $[r(\mathbf{w}_{i,1} + m\mathbf{w}_{i,2})]_2 = \mathbf{d}_{i,m}$. If no party deviates, committing to those values and then multiplying the results produces the right output.
To catch malicious users, without adding a linear number of NIZK arguments, we device a series of ad-hoc checks. First of all calling $k_{j,1}$ and $k_{j,2}$ the shares of the public key returned by P_j, we ask P_j to produce a UC proof of knowledge of $\mathbf{w}_{j,1}, \mathbf{w}_{j,2}$ such that $k_{j,b} = \mathbf{k}^{\mathbf{w}_{j,b}}$. Next each player perform the test

$$e([\mathbf{a}]_1\,, \mathbf{d}_{j,m}) \overset{?}{=} e(k_{j,1}k_{j,2}^m, [r]_2).$$

Although this condition does not implies $\mathbf{d}_{j,m} = [r\mathbf{w}_{j,1} + rm\mathbf{w}_{j,2}]_2$, if this is not the case with significant probability then the adversary is also capable of braking DDH over $\mathbb{G}_1$. To see this observe that extracting $\mathbf{w}_{j,b}$ one can set

$$\mathbf{t}_m = \mathbf{d}_{j,m}\,[-r\mathbf{w}_{j,1} - rm\mathbf{w}_{j,2}]_2 \quad \Rightarrow \quad e(\mathbf{k}, \mathbf{t}_m) = 1.$$

Party $P^{(i)}_{\kappa,\mathsf{SSLE}}$ realising $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$:

```
Initialize $C, D \leftarrow \varnothing$. Send setup to $\mathcal{F}_{\mathsf{Setup}}$ and wait for the reply $(\mathsf{input}, \mathsf{pp}, \mathsf{sp}_i)$. Parse $\mathsf{pp} = \mathsf{mpk}, g, h, k_0, \dots, k_{N-1}$, with $\mathsf{mpk} = [\mathbf{a}]_1, [\mathbf{a}^\top \mathbf{u}]_1, [\mathbf{a}^\top \mathbf{w}_1]_1, [\mathbf{a}^\top \mathbf{w}_2]_1$ together with bilinear groups description, and $\mathsf{sp}_i = \mathsf{sk}_i, f(i)$. Upon receiving:
1 : $(\mathsf{elect}, eid)$: Send $(\mathsf{toss}, eid)$ to $\mathcal{F}^{\kappa,D}_{\mathsf{CT}}$ and set $\mathsf{cnt} \leftarrow 1$
2 : $(\mathsf{tossed}, eid, Q)$ from $\mathcal{F}^{\kappa,D}_{\mathsf{CT}}$: if $i \in Q$,
3 :     $s_i, r_i \xleftarrow{\$} \mathbb{F}_q$, $m_i \xleftarrow{\$} [N]$, $\widetilde{G}_i \leftarrow g^{s_i}$, $\widetilde{H}_i \leftarrow h^{s_i}$, $R_i \leftarrow g^{r_i}$, $\widetilde{c}_{1,i} \leftarrow h^{r_i}\,[m_i \mathbf{a}^\top \mathbf{u}]_1$
4 :     $\pi^i_{\mathsf{Rng}} \xleftarrow{\$} \mathsf{NIZK.P}_{\mathsf{Rng}}\left(g, h, [\mathbf{a}^\top \mathbf{u}]_1, R_i, \widetilde{c}_{1,i}, [N], r_i, m_i\right)$
5 :     $\pi^{0,i}_{\mathsf{DDH}} \xleftarrow{\$} \mathsf{NIZK.P}_{\mathsf{DDH}}\left((g,h), (\widetilde{G}_i, \widetilde{H}_i), s_i\right)$
6 :     Send $\left(\mathsf{commit}, eid||0, \widetilde{G}_i, \widetilde{H}_i, R_i, \widetilde{c}_{1,i}, \pi^i_{\mathsf{Rng}}, \pi^{0,i}_{\mathsf{DDH}}\right)$ to $\mathcal{F}_{\mathsf{Com}}$
7 : $(\mathsf{error})$ from $P_j$ for some $j \in Q$:
8 :     Set $Q \leftarrow Q \setminus \{j\}$ and $D \leftarrow D \cup \{j\}$. If $Q = \varnothing$ send $(\mathsf{toss}, eid||\perp)$ to $\mathcal{F}^{\kappa,D}_{\mathsf{CT}}$
9 : $(\mathsf{recepit}, eid||0, j)$ from $\mathcal{F}_{\mathsf{Com}}$ $\forall j \in Q$: Send $(\mathsf{open}, eid||0)$ to $\mathcal{F}_{\mathsf{Com}}$
10 : $(\mathsf{decom}, eid||0, j, \widetilde{G}_j, \widetilde{H}_j, R_j, \widetilde{c}_{1,j}, \pi^j_{\mathsf{Rng}}.\pi^{0,j}_{\mathsf{DDH}})$ for all $j \in Q$:
11 :     Compute: $\widetilde{G} \leftarrow \prod_{j\in Q} \widetilde{G}_j$,   $\widetilde{H} \leftarrow \prod_{j\in Q} \widetilde{H}_j$,   $\widetilde{c}_1 \leftarrow [\mathbf{a}^\top \mathbf{w}_1]_1 \prod_{j\in Q} \widetilde{c}_{1,j}$,
12 :         $G_i \leftarrow \widetilde{G}^{r_i}$, $H_i \leftarrow \widetilde{H}^{r_i}$, $\mathbf{c}_{0,i} \leftarrow [s_i \mathbf{a}]_1$, $c_{1,i} \leftarrow \widetilde{c}_1^{s_i}$, $c_{2,i} \leftarrow [s_i \mathbf{a}^\top (\mathbf{w}_2 - \mathbf{u})]_1$
13 :         $\pi^{1,i}_{\mathsf{DDH}} \leftarrow \mathsf{NIZK.P}_{\mathsf{DDH}}\left(\left(g, [\mathbf{a}]_1, \widetilde{c}_1, [\mathbf{a}^\top(\mathbf{w}_2 - \mathbf{u})]_1\right), (\widetilde{G}_i, \mathbf{c}_{0,i}, c_{1,i}, c_{2,i}), s_i\right)$
14 :         $\pi^{2,i}_{\mathsf{DDH}} \leftarrow \mathsf{NIZK.P}_{\mathsf{DDH}}\left((g, \widetilde{G}), (R_i, G_i), r_i\right)$ ,
15 :         $\pi^{3,i}_{\mathsf{DDH}} \leftarrow \mathsf{NIZK.P}_{\mathsf{DDH}}\left((g, \widetilde{H}), (R_i, H_i), r_i\right)$
16 :     Send $(\mathsf{commit}, eid||\mathsf{cnt}, H_i, \pi^{3,i}_{\mathsf{DDH}})$ to $\mathcal{F}_{\mathsf{Com}}$
17 :     Broadcast $\left(\mathsf{msg}, eid||\mathsf{cnt}, G_i, \mathbf{c}_{0,i}, c_{1,i}, c_{2,i}, \pi^{1,i}_{\mathsf{DDH}}, \pi^{2,i}_{\mathsf{DDH}}\right)$
```

Fig. 4: Protocol $P^{(i)}_{\kappa,\mathsf{SSSLE}}$, first part

If $\mathbf{t}_m$ is a non trivial vector it allows to decide whether $\mathbf{k}'$ is proportional to $\mathbf{k}$ or random by testing $e(\mathbf{k}', \mathbf{t}_m) = 1$. In the first case the condition is always satisfied while in the second this happens with probability smaller than q^{-1}.
One drawback of this test is that it make parties acts like decryption oracle. For this reason we require also a NIZK PoK of the randomness used to encrypt. By recycling randomness, that is safe in ElGamal encryption with several recipients, we reduce the number of these NIZKs from κN to κ per dealer.
Finally for technical reasons we sample $\bar{\mathbf{w}}_1, \bar{\mathbf{w}}_2 \xleftarrow{\$} \mathbb{F}_q^2$ through the random beacon and multiply the master public key's last components by $\mathbf{k}^{\bar{\mathbf{w}}_b}$ and the secret key's first term $\mathbf{d}_m$ by $[r\bar{\mathbf{w}}_1 + rm\bar{\mathbf{w}}_2]_2$. This helps our simulator in removing the contributions of malicious parties.

To summarize our protocol for setup (see Figure 6) uses:

- NIZKs for relations $\mathcal{R}_{\mathsf{DDH}}$ and $\mathcal{R}_{\mathsf{Lin}}$.
- A global random oracle $\mathcal{H}$, which is also used to realize the setup assumption of the above NIZKs.
- The coin tossing functionality $\mathcal{F}^{\lambda,D,1}_{\mathsf{CT}}$ which, as in the previous sections, returns a random subset $Q \subseteq [N] \setminus D$ plus a string $R \sim U(\{0,1\}^\lambda)$.
- Another coin tossing functionality, $\mathcal{F}^{n,\mathbb{F}_q}_{\mathsf{CT}}$, which returns a uniformly distributed vector of $\mathbb{F}_q^n$.

Party $P^{(i)}_{\kappa,\mathsf{SSLE}}$ realising $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ (second half):

```
18 : $(\mathsf{recepit}, eid||\mathsf{cnt}, j)$, $(\mathsf{msg}, eid||\mathsf{cnt}, G_j, \mathbf{c}_{0,j}, c_{1,j}, c_{2,j})$ or $\mathsf{error}$ from $P_j$, $\forall j \in Q$:
19 :   If the set $M$ of parties that sent error or got $\pi^{1,j}_{\mathsf{DDH}}$, $\pi^{2,j}_{\mathsf{DDH}}$ rejected is non-empty:
20 :     $Q \leftarrow Q \setminus M$, $D \leftarrow D \cup M$, $\mathsf{cnt} \leftarrow \mathsf{cnt} + 1$ and restart from line 11
21 :   Else: send $(\mathsf{open}, eid||\mathsf{cnt})$ to $\mathcal{F}_{\mathsf{Com}}$
22 : $(\mathsf{decom}, eid||\mathsf{cnt}, j, H_j, \pi^{3,j}_{\mathsf{DDH}})$ from $\mathcal{F}_{\mathsf{Com}}$ or $\mathsf{error}$ from $P_j$ for all $j \in Q$:
23 :   If the set $A$ of $P_j$ who sent error or an invalid $\pi^{3,j}_{\mathsf{DDH}}$ is empty set $H \leftarrow \prod_{j \in Q} H_j$;
24 :   Else: $G \leftarrow \prod_{j \in A} G_j$, $K_i \leftarrow G^{f(i)}$,  $\pi^{4,i}_{\mathsf{DDH}} \leftarrow \mathsf{NIZK.P_{DDH}}((g, G), (k_i, K_i), f(i))$
25 :     Send $(\mathsf{recon}, K_i, \pi^{4,i}_{\mathsf{DDH}})$; Upon receiving $(\mathsf{recon}, K_j, \pi^{4,j}_{\mathsf{DDH}})$:
26 :     If the set $R$ of $P_j$ who sent valid $\pi^{4,j}_{\mathsf{DDH}}$ is s.t. $|R| = t + 1$:
27 :     Reconstruct $H_A \leftarrow \prod_{k \in R} K_k^{\lambda_k}$ with $\lambda_k$ the Lagrange coefficients to get $f(-1)$
28 :     Set $H \leftarrow H_A \cdot \prod_{j \in Q \setminus A} H_j$
29 :   Set $\mathbf{c}_0 \leftarrow \prod_{j \in Q} \mathbf{c}_{0,j}$, $c_1 \leftarrow H^{-1} \prod_{j \in Q} c_{1,j}$, $c_2 \leftarrow \prod_{j \in Q} c_{2,j}$ and $c \leftarrow (\mathbf{c}_0, c_1, c_2)$
30 :   Store $C \leftarrow C \cup \{(eid, c)\}$
31 :   If $\exists m \in \phi^{-1}(i)$ such that $1 \leftarrow \mathsf{FE.Dec}(c, (1, m), \mathsf{mpk}, \mathsf{sk}_{i,m})$: Return $(\mathsf{won}, eid)$
32 :   Else: Return $(\mathsf{lost}, eid)$
33 : $(\mathsf{reveal}, eid)$:
34 :   If $(eid, c) \in C$ and $\exists m \in \phi^{-1}(i)$ such that $1 \leftarrow \mathsf{FE.Dec}(c, (1, m), \mathsf{mpk}, \mathsf{sk}_{i,m})$:
35 :     Prove $\pi \xleftarrow{\$} \mathsf{NIZK.P_{Dec}}(\mathsf{mpk}, c, (1, m), \mathsf{sk}_{i,m})$ and broadcast $(\mathsf{claim}, eid, \pi, m)$
36 : $(\mathsf{claim}, eid, \pi, m)$ from $P_j$ with $(eid, c) \in C$:
37 :   If $\phi(m) = j$ and $\pi$ is accepted return $(\mathsf{result}, eid, j)$ otherwise $(\mathsf{rejected}, eid, j)$
```

Fig. 5: Protocol $P^{(i)}_{\kappa,\mathsf{SSLE}}$, second part

The VSS Functionality $\mathcal{F}_{\mathsf{VSS}}$
Sample $g \xleftarrow{\$} \mathbb{G}_1$, $f \xleftarrow{\$} \mathbb{F}_q[x]_{<t}$ and compute $s_i = g^{s_i}$.
Upon receiving $(\mathsf{share\_request}, sid)$ from P_i send $(\mathsf{share}, sid, g, (s_j)_{j=0}^{N-1}, f(i))$

Finally we keep the conventions on $S = [\kappa N]$ and $\phi : S \to [N]$ the reduction modulo N. Moreover we call $\psi : S \to [\kappa]$ the floored division by N i.e. such that $m = N\psi(m) + \phi(m)$ for all $m \in S$.

Theorem 3 *The protocol $\{P^{(i)}_{\mathsf{Setup}} : i \in [N]\}$ securely realises $\mathcal{F}_{\mathsf{Setup}}$ in the $(\mathcal{F}_{\mathsf{CT}}, \mathcal{F}_{\mathsf{Czk}}, \mathcal{F}_{\mathsf{Com}}, \mathcal{F}_{\mathsf{VSS}})$-hybrid model under the* SXDH *assumption for the class of* PPT *environments $\mathcal{Z}$ that statically corrupt up to $\lfloor N/2 \rfloor$ players.*

6 Efficiency considerations

In this section we provide a detailed analysis of the communication efficiency of our protocol, for both the setup and the election phases.

In each case we also provide a concrete estimation, considering an instantiation of the protocol with $N = 2^{14}$ users, and 128 bits of security (so that we can assume that the bitsize of elements of $\mathbb{F}_q, \mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ is 256, 256, 512 and 3072 respectively, and a commitment is 256 bits long). Also

Party $P^{(i)}_{\mathsf{Setup}}$ realising $\mathcal{F}_{\mathsf{Setup}}$:

```
Initialize $D \leftarrow \varnothing,\ \mathbf{x}_i \xleftarrow{\$} \mathbb{F}_q^2,\ \widehat{\mathbf{h}}_i \leftarrow [\mathbf{x}_i]_2$ and prove $\pi^i_{\mathsf{Lin}} \leftarrow \mathsf{NIZK.P_{Lin}}(g_2, \widehat{\mathbf{h}}_i, \mathbf{x}_i)$. Send
$(\mathsf{commit}, \perp, \widehat{\mathbf{h}}_i, \pi^i_{\mathsf{Lin}})$ to $\mathcal{F}_{\mathsf{Com}}$. Upon receiving $(\mathsf{recepit}, \perp, j)$ or error for all $j \in [N] \setminus \{i\}$,
send $(\mathsf{open}, \perp)$ to $\mathcal{F}_{\mathsf{Com}}$ and wait for $(\mathsf{decom}, \perp, j, \widehat{\mathbf{h}}_j, \pi^j_{\mathsf{Lin}})$ or error for all $P_j$. Add to $D$
players who sent error or got $\pi^j_{\mathsf{Lin}}$ rejected. Upon receiving:
1 : $(\mathsf{setup}, sid)$: Send $(\mathsf{share\_request}, sid)$ to $\mathcal{F}_{\mathsf{VSS}}$; wait $(\mathsf{share}, sid, g, (s_j)_{j=0}^{N-1}, f(i))$.
2 :   Send $(\mathsf{toss}, sid)$ to $\mathcal{F}^{\lambda,D,1}_{\mathsf{CT}}$
3 : $(\mathsf{tossed}, sid, Q, R)$ from $\mathcal{F}^{\lambda,D,1}_{\mathsf{CT}}$: If $i \in Q$:
4 :   Compute $\mathbf{k}, k_0, (d_m)_{m \in S} \leftarrow \mathcal{H}(R)$ with $\mathbf{k} \in \mathbb{G}_1^2$, $k_0 \in \mathbb{G}_1$ and each $d_m \in \mathbb{G}_2$
5 :   Sample $\mathbf{w}_{i,1}, \mathbf{w}_{i,2} \xleftarrow{\$} \mathbb{F}_q^2,\ r_{i,\psi(m)} \xleftarrow{\$} \mathbb{F}_q$; for all $m \in \phi^{-1}([N] \setminus D)$ compute:
6 :     $k_{i,1} \leftarrow \mathbf{k}^{\mathbf{w}_{i,1}}, \quad k_{i,2} \leftarrow \mathbf{k}^{\mathbf{w}_{i,2}}, \quad \mathbf{d}_{i,m} \leftarrow [\mathbf{w}_{1,i} + m\mathbf{w}_{2,i}]_{d_m}$
7 :     $G_{i,\psi(m)} \leftarrow \left[r_{i,\psi(m)}\right]_2, \quad C_{i,m} \leftarrow \widehat{\mathbf{h}}^{r_{i,\psi(m)}}_{\phi(m)} \cdot \mathbf{d}_{i,m}$
8 :     $\pi^{j,\psi(m)}_{\mathsf{DDH}} \leftarrow \mathsf{NIZK.P_{DDH}}\left(g_2, G_{i,\psi(m)}, r_{i,\psi(m)}\right)$
9 :   Send $(\mathsf{commit}, sid, G_{i,\psi(m)}, C_{i,m}, \pi^{i,\psi(m)}_{\mathsf{DDH}})$ to $\mathcal{F}_{\mathsf{Com}}$
10 :   Send $(\mathsf{prove}, sid, \mathbf{k}^\top, (k_{i,1}, k_{i,2}), (\mathbf{w}_{i,1}, \mathbf{w}_{i,2})^\top)$ to $\mathcal{F}^{\mathsf{Lin}}_{\mathsf{Czk}}$
11 : $(\mathsf{recepit}, sid, j)$ from both $\mathcal{F}_{\mathsf{Com}}$ and $\mathcal{F}^{\mathsf{Lin}}_{\mathsf{Czk}}$ or error for all $j \in Q$:
12 :   For all $P_j$ that sent error, add $D \leftarrow D \cup \{j\}$ and remove $Q \leftarrow Q \setminus \{j\}$
13 :   Send $(\mathsf{open}, sid)$ to $\mathcal{F}_{\mathsf{Com}}$ and $\mathcal{F}^{\mathsf{Lin}}_{\mathsf{Czk}}$
14 : $(\mathsf{decom}, sid, j, G_{j,\psi(m)}, C_{j,m}, \pi^{j,\psi(m)}_{\mathsf{DDH}})$, $(\mathsf{proof}, sid, j, \mathbf{k}, (k_{j,1}, k_{j,2}))$ or error, $\forall \in Q$:
15 :   Remove from $Q$ and add to $D$ parties that sent error or incorrect $\pi^{j,\psi(m)}_{\mathsf{DDH}}$
16 :   Send $(\mathsf{toss}, sid)$ to $\mathcal{F}^{4,\mathbb{F}_q}_{\mathsf{CT}}$
17 :   **For** all $j \in Q$, $m \in \phi^{-1}(i)$: Decrypt $\mathbf{d}_{j,m} \leftarrow C_{j,m} \cdot G^{-\mathbf{x}_i}_{j,\psi(m)}$
18 :     **If** $e(\mathbf{k}, \mathbf{d}_{j,m}) \neq e(k_{j,1}k^m_{j,2}, d_m)$ for some $m \in \phi^{-1}(i)$:
19 :       Prove $\pi^{j,m}_{\mathsf{Lin}} \leftarrow \mathsf{NIZK.P_{Lin}}\left(\left(g_2, G_{j,\psi(m)}\right), \left(\widehat{\mathbf{h}}_i, C_{j,m}\mathbf{d}^{-1}_{j,m}\right), \mathbf{x}_i^\top\right)$
20 :       Broadcast $\left(\mathsf{complain}, \pi^{j,m}_{\mathsf{Lin}}, \mathbf{d}_{j,m}, j, m\right)$
21 :     **Else**: Broadcast $(\mathsf{accept}, j)$.
22 : $(\mathsf{complain}, \pi^{j',m}_{\mathsf{Lin}}, \mathbf{d}_{j,m}, j, m)$ or error from $P_{j'}$:
23 :   If $P_{j'}$ sent error, $\pi^{j,m}_{\mathsf{Lin}}$ is rejected or $e(\mathbf{k}, \mathbf{d}_{j,m}) \neq e(k_{1,j}k^m_{2,j}, d_m)$: add $D \leftarrow D \cup \{j'\}$
24 :   Otherwise add $D \leftarrow D \cup \{j\}$ and remove $Q \leftarrow Q \setminus \{j\}$
25 : $(\mathsf{accept}, j)$ for all $j \in Q$ and $(\mathsf{tossed}, sid, \bar{\mathbf{w}}_1, \bar{\mathbf{w}}_2)$ from $\mathcal{F}^{4,\mathbb{F}_q}_{\mathsf{CT}}$:
26 :   For all $m \in \phi^{-1}(i)$ compute : $k_1 \leftarrow \mathbf{k}^{\bar{\mathbf{w}}_1} \cdot \prod_{j \in Q} k_{j,1}, \quad k_2 \leftarrow \mathbf{k}^{\bar{\mathbf{w}}_2} \cdot \prod_{j \in Q} k_{j,2}$
27 :     $\mathbf{d}_m \leftarrow [\bar{\mathbf{w}}_1 + m\bar{\mathbf{w}}_2]_{d_m} \cdot \prod_{j \in Q} \mathbf{d}_{j,m}, \quad \sigma_m \xleftarrow{\$} \mathbb{F}_q, \quad \mathsf{sk}_{i,m} \leftarrow (\mathbf{d}^{\sigma_m}_m, d^{\sigma_m}_m)$
28 :     $\mathsf{sk}_i \leftarrow (\mathsf{sk}_{i,m})_{m \in \phi^{-1}(i)}, \quad \mathsf{mpk} \leftarrow (\mathbf{k}, k_0, k_1, k_2)$
29 :   Call $\mathsf{pp} \leftarrow (\mathsf{mpk}, g, h, (s_j)_{j=0}^{N-1})$, $\mathsf{sp}_i \leftarrow (\mathsf{sk}_i, f(i))$ and return $(\mathsf{input}, sid, \mathsf{pp}, \mathsf{sp}_i)$
```

Fig. 6: Realisation of the ideal $\mathcal{F}_{\mathsf{Setup}}$ functionality

we set the statistical security parameter for the size of the committee used in the setup phase as $\kappa_s = 80$, while for the election we set it as $\kappa_e = \log N = 14$.

Setup. The rounds and the messages sent in the protocol are as follows.

1. Every player sends one commitment.
2. Every player P_j opens the commitment, sending 4 elements of $\mathbb{G}_2$ and 2 elements of $\mathbb{F}_q$ ($\widehat{\mathbf{h}}_j, \pi^j_{\mathsf{Lin}}$).
3. All the players execute a VSS protocol.
4. All the players invoke the coin tossing functionality.
5. Every player in Q sends two commitments.
6. Every player in Q opens the commitments, sending $\kappa_e \cdot N$ elements of $\mathbb{G}_2^2$ ($\{C_{i,m}\}$), κ_e elements of $\mathbb{G}_2$ ($\{G_{i,\psi(m)}\}$), 2 elements of $\mathbb{G}_1$ ($k_{i,1}, k_{i,2}$), 40 elements of $\mathbb{G}_1$ and 20 of $\mathbb{F}_q$ ($\mathcal{F}^{\mathsf{Lin}}_{\mathsf{Czk}}$ proof), κ_e elements of $\mathbb{G}_2$ and $\mathbb{F}_q$ ($\pi^{i,\psi(m)}_{\mathsf{DDH}}$).
7. Invoke the coin toss functionality. Let A be the set of players who misbehave. Every player may send 2 elements of $\mathbb{G}_2$ ($\mathbf{d}_{j,m}$) and 4 elements of $\mathbb{G}_2$ and 2 elements of $\mathbb{F}_q$ ($\pi^{j,m}_{\mathsf{Lin}}$) for every P_j s.t. $j \in A$.

To evaluate the VSS we assume that every player P_j in the committee samples a polynomial $f_j \xleftarrow{\$} \mathbb{F}_q[x]_{<t}$ and broadcasts $g^{f(i)}$ and an encryption of $f(i)$, that is $N+1$ elements in $\mathbb{G}_1$ and roughly N field elements. So ignoring the cost of the coin tossing functionalities, every player in the committee Q (which has size κ_s) sends in total 3 commitments, 42 elements of $\mathbb{G}_1$, $2\kappa_e(N+1)+4$ elements of $\mathbb{G}_2$, and $22+\kappa_e$ elements of $\mathbb{F}_q$. In addition, in the last round, every player may send 6 elements of $\mathbb{G}_2$ and 2 of $\mathbb{F}_q$ for every player that misbehaved. With the parameters mentioned earlier, each player in the setup committee sends about 29 MB, while any user receive about 480 kB[11] to build its secret key.

Election. The rounds and the messages sent in the preprocessing phase of the election protocol (in case no party misbehaves) are as follows:

1. All the players execute the coin tossing functionality
2. Every player in Q sends one commitment
3. Every player in Q opens the commitments, sending 4 elements of $\mathbb{G}_1$ ($\widetilde{G}_i, \widetilde{H}_i, R_i, \widetilde{c}_{1,i}$), 1 range proof ($\pi^i_{\mathsf{Rng}}$), 1 element of $\mathbb{G}_1$ and 1 of $\mathbb{F}_q$ ($\pi^{0,i}_{\mathsf{DDH}}$).
4. Every player in Q sends one commitment and 5 elements of $\mathbb{G}_1$ ($G_i, \mathbf{c}_{0,i}, c_{1,i}, c_{2,i}$), 2 elements of $\mathbb{G}_1$ and of $\mathbb{F}_q$ ($\pi^{1,i}_{\mathsf{DDH}}, \pi^{2,i}_{\mathsf{DDH}}$).

The online phase instead proceeds in the following way:

1. Every player in Q opens the commitment, sending two elements of $\mathbb{G}_1$ and one of $\mathbb{F}_q$ ($H_i, \pi^{3,i}_{\mathsf{DDH}}$).
2. If one of the messages sent in the above round is incorrect (or the player didn't answer), the honest players post partial decryptions, sending 2 elements of $\mathbb{G}_1$ and one of $\mathbb{F}_q$ ($K_i, \pi^{4,i}_{\mathsf{DDH}}$).
3. The winner sends a proof of correct decryption, which consists of 2 elements of $\mathbb{G}_T$ and 3 elements of $\mathbb{G}_2$.

So, without considering the cost of building the random beacon, every player in the committee Q (of size κ_e) in the processing sends 2 commitments, $15 + 3\log N$ elements of $\mathbb{G}_1$ and $6 + 3\log N$ elements of $\mathbb{F}_q$ (and receives about $\kappa_e\times$ this information from all the other users). Using $\kappa_e = \log N$

[11] And by considering a higher $\kappa_e = 80$ these costs become roughly $5\times$ more.

(resp. $\kappa_e = 80$) as a statistical parameter for the committee size in the election, we have that each player sends about 3 kB and must receives 47 kB (resp. 274 kB). The online phase is quite efficient. In the best case (when all members of Q open the commitment), it involves the sending of a total of 2.3 kB (resp. 8.6 kB). In the worst case, the threshold decryption requires the communication of 789 kB in total. Finally, the winner only needs to send less than 1 kB to claim her victory.

Comparison with the efficient construction from [BEHG20]. To compare our costs with those of the efficient shuffle-based construction from [BEHG20] in the fairest possible way, we consider two possible ways of implementing their protocol. In both variants we distinguish a setup phase and an election phase[12], the difference lies in how the setup is realized. The first variant assumes a preliminary stage where users register their Diffie-Hellman pairs. Next, a committee (of κ_s users) is randomly selected via the random beacon. The committee then proceeds by performing κ_s shuffling sequentially. Using the same parameters as above, in such a setting, each player sends only 1 MB but needs to receive about 85 Mbytes (to store the κ_e lists and check the shufflings). The second variant, which is actually the one explicitly suggested in [BEHG20], consists in each newly registered player performing a shuffling when entering in the system. Then, whenever a leader is elected the latter re-registers with new keys and performs a new shuffle. The disadvantage of this solution is that it requires very high initial costs. Again assuming the same parameters as above, setting up the system from scratch requires every player to receive 8.5 GB.

We stress, however, that in both variants, setup is done once and for all at the beginning of the protocol (i.e. no need to repeat it when the set of participants changes, only one shuffle per new participants is needed).

As per the election phase, both variants induce communication costs of 1 MB (as the winner needs to re-register to be able to take part of future elections).

In conclusion, these figures suggest that our scheme is a better option in settings where many elections, involving a stable set of participants, are expected. This allows to take advantage of the sublinear election costs of the scheme while reducing the impact of the high setup costs. If we compare the total communication cost for 1000 elections, our protocol (including the setup, and 1000 offline and online elections) is still competitive as it entails a total communication sent/received by each player of 80 MB, in contrast to 1.1 GB in [BEHG20].

In highly dynamic scenarios, on the other hand, the shuffling-based solution works better as the higher election costs are mitigated by the benefits of a more flexible setup.

Acknowledgements

Research leading to these results has been partially supported by the Spanish Government under projects SCUM (ref. RTI2018-102043-B-I00), CRYPTOEPIC (refs. EUR2019-103816), and SECURITAS (ref. RED2018-102321-T), by the Madrid Regional Government under project BLOQUES (ref. S2018/TCS-4339), and by a research grant from Nomadic Labs and the Tezos Foundation.

References

ABC$^+$05. M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable Encryption Revisited: Consistency Properties, Relation to Anonymous IBE, and

[12] We stress that the division in setup phase and election phase is not explicitly considered in [BEHG20] and is presented here for the sole sake of clarity

Extensions. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 205–222. Springer, Heidelberg, August 2005.

ADMM14a. M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek. Fair Two-Party Computations via Bitcoin Deposits. In R. Böhme, M. Brenner, T. Moore, and M. Smith, editors, *FC 2014 Workshops*, volume 8438 of *LNCS*, pages 105–121. Springer, Heidelberg, March 2014.

ADMM14b. M. Andrychowicz, S. Dziembowski, D. Malinowski, and L. Mazurek. Secure Multiparty Computations on Bitcoin. In *2014 IEEE Symposium on Security and Privacy*, pages 443–458. IEEE Computer Society Press, May 2014.

AMM18. S. Azouvi, P. McCorry, and S. Meiklejohn. Betting on Blockchain Consensus with Fantomette. *CoRR*, abs/1805.06786, 2018.

BBB$^+$18. B. Bünz, J. Bootle, D. Boneh, A. Poelstra, P. Wuille, and G. 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.

BDD20. C. Baum, B. David, and R. Dowsley. Insured MPC: Efficient Secure Computation with Financial Penalties. In J. Bonneau and N. Heninger, editors, *FC 2020*, volume 12059 of *LNCS*, pages 404–420. Springer, Heidelberg, February 2020.

BDOP04. D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public Key Encryption with Keyword Search. In C. Cachin and J. Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 506–522. Springer, Heidelberg, May 2004.

BEHG20. D. Boneh, S. Eskandarian, L. Hanzlik, and N. Greco. Single Secret Leader Election. Cryptology ePrint Archive, Report 2020/025, 2020. `https://eprint.iacr.org/2020/025`.

BGG$^+$18. D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. M. R. Rasmussen, and A. Sahai. Threshold Cryptosystems from Threshold Fully Homomorphic Encryption. In H. Shacham and A. Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 565–596. Springer, Heidelberg, August 2018.

BGM16. I. Bentov, A. Gabizon, and A. Mizrahi. Cryptocurrencies Without Proof of Work. In J. Clark, S. Meiklejohn, P. Y. A. Ryan, D. S. Wallach, M. Brenner, and K. Rohloff, editors, *FC 2016 Workshops*, volume 9604 of *LNCS*, pages 142–157. Springer, Heidelberg, February 2016.

BK14. I. Bentov and R. Kumaresan. How to Use Bitcoin to Design Fair Protocols. In J. A. Garay and R. Gennaro, editors, *CRYPTO 2014, Part II*, volume 8617 of *LNCS*, pages 421–439. Springer, Heidelberg, August 2014.

BPS16. I. Bentov, R. Pass, and E. Shi. Snow White: Provably Secure Proofs of Stake. Cryptology ePrint Archive, Report 2016/919, 2016. `http://eprint.iacr.org/2016/919`.

BSW11. D. Boneh, A. Sahai, and B. Waters. Functional Encryption: Definitions and Challenges. In Y. Ishai, editor, *TCC 2011*, volume 6597 of *LNCS*, pages 253–273. Springer, Heidelberg, March 2011.

Can01. R. Canetti. Universally Composable Security: A New Paradigm for Cryptographic Protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

CD20. I. Cascudo and B. David. ALBATROSS: Publicly AttestabLe BATched Randomness Based On Secret Sharing. In S. Moriai and H. Wang, editors, *ASIACRYPT 2020, Part III*, volume 12493 of *LNCS*, pages 311–341. Springer, Heidelberg, December 2020.

CDG$^+$18. J. Camenisch, M. Drijvers, T. Gagliardoni, A. Lehmann, and G. Neven. The Wonderful World of Global Random Oracles. In J. B. Nielsen and V. Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 280–312. Springer, Heidelberg, April / May 2018.

CDPW07. R. Canetti, Y. Dodis, R. Pass, and S. Walfish. Universally Composable Security with Global Setup. In S. P. Vadhan, editor, *TCC 2007*, volume 4392 of *LNCS*, pages 61–85. Springer, Heidelberg, February 2007.

CF01. R. Canetti and M. Fischlin. Universally Composable Commitments. In J. Kilian, editor, *CRYPTO 2001*, volume 2139 of *LNCS*, pages 19–40. Springer, Heidelberg, August 2001.

Fis05. M. Fischlin. Communication-Efficient Non-interactive Proofs of Knowledge with Online Extractors. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 152–168. Springer, Heidelberg, August 2005.

FKMV12. S. Faust, M. Kohlweiss, G. A. Marson, and D. Venturi. On the Non-malleability of the Fiat-Shamir Transform. In S. D. Galbraith and M. Nandi, editors, *INDOCRYPT 2012*, volume 7668 of *LNCS*, pages 60–79. Springer, Heidelberg, December 2012.

FLS90. U. Feige, D. Lapidot, and A. 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. S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate Indistinguishability Obfuscation and Functional Encryption for all Circuits. In *54th FOCS*, pages 40–49. IEEE Computer Society Press, October 2013.

GHM$^+$17. Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. 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. C. Ganesh, C. Orlandi, and D. Tschudi. Proof-of-Stake Protocols for Privacy-Aware Blockchains. In Y. Ishai and V. Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 690–719. Springer, Heidelberg, May 2019.

GPS08. S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for Cryptographers. *Discrete Appl. Math.*, 156(16):3113–3121, September 2008.

KB16. R. Kumaresan and I. Bentov. Amortizing Secure Computation with Penalties. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, and S. Halevi, editors, *ACM CCS 2016*, pages 418–429. ACM Press, October 2016.

KKKZ19. T. Kerber, A. Kiayias, M. Kohlweiss, and V. 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.

KMB15. R. Kumaresan, T. Moran, and I. Bentov. How to Use Bitcoin to Play Decentralized Poker. In I. Ray, N. Li, and C. Kruegel, editors, *ACM CCS 2015*, pages 195–206. ACM Press, October 2015.

KSW08. J. Katz, A. Sahai, and B. Waters. Predicate Encryption Supporting Disjunctions, Polynomial Equations, and Inner Products. In N. P. Smart, editor, *EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 146–162. Springer, Heidelberg, April 2008.

KZZ16. A. Kiayias, H.-S. Zhou, and V. Zikas. Fair and Robust Multi-party Computation Using a Global Transaction Ledger. In M. Fischlin and J.-S. Coron, editors, *EUROCRYPT 2016, Part II*, volume 9666 of *LNCS*, pages 705–734. Springer, Heidelberg, May 2016.

Lab19. P. Labs. Secret single-leader election (SSLE), 2019.

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

NR97. M. Naor and O. Reingold. Number-theoretic Constructions of Efficient Pseudo-random Functions. In *38th FOCS*, pages 458–467. IEEE Computer Society Press, October 1997.

O'N10. A. O'Neill. Definitional Issues in Functional Encryption. Cryptology ePrint Archive, Report 2010/556, 2010. `http://eprint.iacr.org/2010/556`.

Sah99. A. Sahai. Non-Malleable Non-Interactive Zero Knowledge and Adaptive Chosen-Ciphertext Security. In *40th FOCS*, pages 543–553. IEEE Computer Society Press, October 1999.

Wee17. H. Wee. Attribute-Hiding Predicate Encryption in Bilinear Groups, Revisited. In Y. Kalai and L. Reyzin, editors, *TCC 2017, Part I*, volume 10677 of *LNCS*, pages 206–233. Springer, Heidelberg, November 2017.

A Statistical Distance

Definition 11. *Given a finite set S and two random variables $x, y \sim S$ we define their statistical distance as*

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

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 $a \in A$, $\Pr\left[x_{|A} = a\right] = \Pr[x = a|A]$. In the rest of this subsection we list some properties of the statistical distance we use in the last proofs.

Proposition 1 *Let $\mathbb{G}$ be a finite group $g, h \sim \mathbb{G}_1$ be statistically independent and $u \sim U(\mathbb{G}_1)$ then*

$$\Delta(gh, u) \leq \Delta(g, u).$$

Proposition 2 *Given S_1, S_2 two sets, $x, y \sim S_1$ and $f : S_1 \to S_2$ any function then $\Delta(f(x), f(y)) \leq \Delta(x, y)$.*

Proposition 3 *Given two set S_1, S_2 and $f : S_1 \to S_2$ a bijection, if $x \sim U(S_1)$ then $f(x) \sim U(S_2)$*

Proposition 4 *Given a finite set S and $x, y \sim S$, then for any subset $A \subseteq S$*

$$\left|\Pr[x \in A] - \Pr[y \in A]\right| \ \leq \ \Delta(x,y).$$

The next proposition 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 5 *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 Sigma Protocols

B.1 The Decryption Relation

In this section we provide a sigma protocol inspired by [Mau15] for the relation $\mathcal{R}_{\mathsf{Dec}}$, define in Section 2.7, instantiated for the FE scheme in Figure 2. Even though in our SSLE protocol we use this scheme only with message space of dimension $n = 2$, the protocol provided here works for message space of any dimension and, remarkably, its communication costs always consist of 5 group elements regardless of the dimension n.

Proposition 6 *Protocol 7 satisfies perfect completeness, special soundness and perfect HVZK*

$$\begin{array}{lcl}
\underline{\mathcal{P}_{\mathsf{Dec}}(\mathsf{mpk}, c, \mathbf{y}, \mathsf{sk}):} & & \underline{\mathcal{V}_{\mathsf{Dec}}(\mathsf{mpk}, c, \mathbf{y}):} \\
\text{Parse } \mathsf{mpk} = (\mathbf{k}, k^*, k_i)_{i=1}^n, & & \text{Parse } \mathsf{mpk} = (\mathbf{k}, k^*, k_i)_{i=1}^n, \\
\quad c = (\mathbf{c}, c_i)_{i=1}^n,\ \mathbf{y} = (y_i)_{i=1}^n, & & \quad c = (\mathbf{c}, c_i)_{i=1}^n,\ \mathbf{y} = (y_i)_{i=1}^n, \\
\quad \mathsf{sk} = (\mathbf{d}, d) & & \\
\mathbf{v} \leftarrow^{\$} \mathbb{G}_2^2, \quad \rho \leftarrow^{\$} \mathbb{F}_q & & \\
\widehat{\mathbf{d}} \leftarrow \mathbf{d}^{\rho}, \quad \widehat{d} \leftarrow d^{\rho} & & \\
T_0 \leftarrow e(\mathbf{k}, \mathbf{v}) & & \\
T_1 \leftarrow e(\mathbf{c}, \mathbf{v}) & \xrightarrow{\widehat{d}, T_0, T_1} & \\
 & \xleftarrow{r} & r \leftarrow^{\$} \mathbb{F}_q \\
\mathbf{z} \leftarrow \widehat{\mathbf{d}}^r \cdot \mathbf{v} & \xrightarrow{\mathbf{z}} & \\
 & & \text{Check:} \\
 & & \quad \widehat{d} \neq 1_{\mathbb{G}_2} \\
 & & \quad e(\mathbf{k}, \mathbf{z}) \stackrel{?}{=} T_0 \cdot e(k_1^{y_1}, \dots, k_n^{y_n}, \widehat{d})^r \\
 & & \quad e(\mathbf{c}, \mathbf{z}) \stackrel{?}{=} T_1 \cdot e(c_1^{y_1}, \dots, c_n^{y_n}, \widehat{d})^r
\end{array}$$

Fig. 7: Sigma protocol for the $\mathcal{R}_{\mathsf{Dec}}$ relation

Proof. **Completeness:** Given $(\mathsf{mpk}, c, \mathbf{y}, \mathsf{sk}) \in \mathcal{R}_{\mathsf{Dec}}$ by construction $(\mathsf{mpk}, \mathbf{y}, \mathsf{sk}) \in \mathcal{L}_{\mathsf{key}}$. Parsing $\mathsf{mpk} = (\mathbf{k}, k^*, k_1, \dots, k_n)$ and $\mathsf{sk} = (\mathbf{d}, d)$ with

$$\mathbf{k} = [\mathbf{a}]_1, \quad k^* = \left[\mathbf{a}^\top \mathbf{u}\right]_1, \quad k_i = \left[\mathbf{a}^\top \mathbf{w}_i\right]_1$$

we have that the vector $c' = (\mathbf{k}, k_1, \dots, k_n)$ is the encryption of $\mathbf{0}$ with randomness 1. From the definition of $\mathcal{L}_{\mathsf{key}}$, as $\mathbf{y}^\top \mathbf{0} = 0$ we deduce that $\mathsf{FE.Dec}(c', \mathbf{y}, \mathsf{mpk}, \mathsf{sk}) = 1$ and $d \neq 1$, while by definition of $\mathcal{R}_{\mathsf{Dec}}$ we get $\mathsf{FE.Dec}(c, \mathbf{y}, \mathsf{mpk}, \mathsf{sk}) = 1$ and in particular

$$\begin{cases} e(\mathbf{k}, \mathbf{d}) = e(k_1^{y_1} \cdot \ldots \cdot k_n^{y_n}, d) \\ e(\mathbf{c}, \mathbf{d}) = e(c_1^{y_1} \cdot \ldots \cdot c_n^{y_n}, d) \end{cases} \Rightarrow \begin{cases} e(\mathbf{k}, \mathbf{d}^{\rho r} \mathbf{v}) = e(\mathbf{k}, \mathbf{v}) \cdot e(k_1^{y_1} \cdot \ldots \cdot k_n^{y_n}, d^{\rho})^r \\ e(\mathbf{c}, \mathbf{d}^{\rho r} \mathbf{v}) = e(\mathbf{c}, \mathbf{v}) \cdot e(c_1^{y_1} \cdot \ldots \cdot c_n^{y_n}, d^{\rho})^r \end{cases}$$

Special Soundness: Given $(\widehat{d}, T_0, T_1, r_1, \mathbf{z}_1)$ and $(\widehat{d}, T_0, T_1, r_2, \mathbf{z}_2)$ two accepting transcripts, calling $\widehat{\mathbf{d}} = (\mathbf{z}_1 \cdot \mathbf{z}_2^{-1})^{(r_1 - r_2)^{-1}}$ we have that

$$\begin{cases} e(\mathbf{k}, \widehat{\mathbf{d}}) = e(k_1^{y_1} \cdot \ldots \cdot k_n^{y_n}, \widehat{d}) \\ e(\mathbf{c}, \widehat{\mathbf{d}}) = e(c_1^{y_1} \cdot \ldots \cdot c_n^{y_n}, \widehat{d}). \end{cases}$$

From the second line $\mathsf{sk} = (\widehat{\mathbf{d}}, \widehat{d})$ decrypts c, that is $\mathsf{FE.Dec}(c, \mathbf{y}, \mathsf{sk}, \mathsf{mpk}) = 1$. We show now that the first equation implies $(\mathsf{mpk}, \mathbf{y}, \mathsf{sk}) \in \mathcal{L}_{\mathsf{key}}$. Fixed $\mathsf{mpk} = (\mathbf{k}, k^*, k_1, \dots, k_n)$ with

$$\mathbf{k} = [\mathbf{a}]_1, \quad k^* = \left[\mathbf{a}^\top \mathbf{u}\right]_1, \quad k_i = \left[\mathbf{a}^\top \mathbf{w}_i\right]_1$$

and $\widehat{\mathbf{d}} = [\mathbf{x}]_2$, $\widehat{d} = [\sigma]_2$, from the first equation

$$\mathbf{a}^\top \mathbf{x} = \mathbf{a}^\top \sum_{i=1}^n \sigma y_i \mathbf{w}_i$$

Unfortunately this does not imply $\mathbf{x} = \sum_{i=1}^{n} \sigma y_i \mathbf{w}_i$, but is enough to conclude the proof. Indeed let $(\bar{\mathbf{c}}, \bar{c}_1, \ldots, \bar{c}_n)$ be the encryption of a vector $\mathbf{m}$ with randomness $s \neq 0$. By construction

$$\bar{\mathbf{c}} = [s\mathbf{a}]_1, \quad \bar{c}_i = [s\mathbf{a}(m_i\mathbf{u} + \mathbf{w}_i)]_1 \quad \Rightarrow \quad e(\bar{\mathbf{c}}, \widehat{\mathbf{d}}) \cdot e(\bar{c}_1^{y_1} \cdot \ldots \cdot \bar{c}_n^{y_n}, \widehat{d})^{-1} = \\ = \Big[s\mathbf{a}^\top + s\mathbf{a}^\top \sum\nolimits_{i=1}^{n} \sigma y_i m_i \mathbf{u} - s\mathbf{a}^\top \sum\nolimits_{i=1}^{n} \sigma y_i \mathbf{w}_i\Big]_T = \Big[s(\mathbf{a}^\top\mathbf{u})(\mathbf{y}^\top\mathbf{m})\Big]_T .$$

Since $s \neq 0$ and $\mathbf{a}^\top\mathbf{u} \neq 0$, the last term is zero if and only if $\mathbf{y}^\top\mathbf{m} = 0$, that is the thesis.

HVZK: Below we provide the description of a simulator $\mathcal{S}_{\mathsf{Dec}}$ that produces an accepting transcript with the right distribution.
Given a tuple $(\mathsf{mpk}, c, \mathbf{y}, \mathsf{sk}) \in \mathcal{R}_{\mathsf{Dec}}$ we show that the statistical distance between $(\widehat{d}, T_0, T_1, r, \mathbf{z})$ and

Simulator $\mathcal{S}_{\mathsf{Dec}}(\mathsf{mpk}, c, \mathbf{y})$:

```
1 : Parse $\mathsf{mpk} = (\mathbf{k}, k^*, k_1, \ldots, k_n)$,   $c = (\mathbf{c}, c_1, \ldots, c_n)$,   $\mathbf{y} = (y_1, \ldots, y_n)$
2 : Sample $r' \leftarrow^{\$} \mathbb{F}_q$,   $\widehat{d'} \leftarrow^{\$} \mathbb{G}_2$,   $\mathbf{z}' \leftarrow^{\$} \mathbb{G}_2^2$
3 : $T_0' \leftarrow e(\mathbf{k}, \mathbf{z}') \cdot e(k_1^{y_1} \cdot \ldots \cdot k_n^{y_n}, \widehat{d'})^{-r}$,   $T_1' \leftarrow e(\mathbf{c}, \mathbf{z}') \cdot e(c_1^{y_1} \cdot \ldots \cdot c_n^{y_n}, \widehat{d'})^{-r}$
4 : Return $(\widehat{d'}, T_0', T_1', r', \mathbf{z}')$
```

$(\widehat{d'}, T_0', T_1', r', \mathbf{z}')$ is zero, where the first one is the transcript generated by $\mathcal{P}_{\mathsf{Dec}}$, $\mathcal{V}_{\mathsf{Dec}}$ and the second one is the output of $\mathcal{S}_{\mathsf{Dec}}$. We begin observing that $(\mathbf{z}, \widehat{d}, r) \sim U(\mathbb{G}_2^3 \times \mathbb{F}_q)$ and $(\mathbf{z}, \widehat{d'}, r') \sim U(\mathbb{G}_2^3 \times \mathbb{F}_q)$. This is true in the first case because $\mathbf{v}$, ρ and r are uniformly and independently distributed, in the second one by construction. Consequently they have statistical distance 0 and by Proposition 5 we only need to show that $\forall \mathbf{z}_0, \widehat{d}_0, r_0$, upon conditioning on $\widehat{d} = \widehat{d}_0 = \widehat{d'}$, $\mathbf{z} = \mathbf{z}_0 = \mathbf{z}'$ and $r = r_0 = r'$, the vectors $(T_0, T_1), (T_0', T_1')$ have statistical distance 0.
In the real protocol $\mathbf{z} = \mathbf{z}_0$ implies $\mathbf{v} = \mathbf{z}_0 \mathbf{d}^{-\rho r_0}$ and in particular

$$\begin{aligned} T_0 &= e(\mathbf{k}, \mathbf{v}) = e(\mathbf{k}, \mathbf{z}_0) \cdot e(\mathbf{k}, \mathbf{d}^\rho)^{-r_0} = \\ &= e(\mathbf{k}, \mathbf{z}_0) \cdot e(k_1^{y_1} \cdot \ldots \cdot k_n^{y_n}, d^\rho)^{-r_0} = \\ &= e(\mathbf{k}, \mathbf{z}_0) \cdot e(k_1^{y_1} \cdot \ldots \cdot k_n^{y_n}, \widehat{d}_0)^{-r_0} = T_0'. \end{aligned}$$

Analogously $T_1 = T_1'$ and therefore $\Delta((T_0, T_1), (T_0', T_1')) = 0$.

B.2 Proof of Range

We now provide a sigma protocol that proves, for uniformly random g, h, k provided in the CRS, that given u, v there exist $r \in \mathbb{F}_q$ and $m \in [2^\mu]$ such that $u = g^r$ and $v = h^r k^m$. We begin by splitting this statement in two easier ones to prove, that is

- $\exists r', m' \in \mathbb{F}_q \;:\; u = g^{r'},\; v = h^{r'} k^{m'}$
- $\exists r \in \mathbb{F}_q,\; m \in [2^\nu] \;:\; v = h^r k^m$

The first relation can be proven with a proof system for $\mathcal{R}_{\mathsf{Lin}}$ because it is equivalent to showing the existence of $r', m' \in \mathbb{F}_q$ such that

$$\begin{bmatrix} g & 1 \\ h & k \end{bmatrix} \cdot \begin{pmatrix} r' \\ m' \end{pmatrix} = \begin{bmatrix} u \\ v \end{bmatrix}$$

To prove the second one we use the folklore trick of committing to m's bits b_i with a Pedersen commitment and prove that each of them contains indeed a bit and $m = \sum_{i=0}^{\mu-1} b_i 2^i$. More in detail, sampling $r_i \leftarrow^{\$} \mathbb{F}_q$ for every i, one can send $c_i = h^{r_i} k^{b_i}$ and then show that

- $\exists r_i \in \mathbb{F}_q,\ b_i \in \{0,1\}\ :\ c_i = h^{r_i} k^{b_i}$
- $\exists \rho \in \mathbb{F}_q,\ h^{\rho} \cdot \prod_{i=0}^{\mu-1} c_i^{2^i} = v$

To prove the first one, a known method is to show that the prover can open c_i both to b_i and b_i^2. Since Pedersen commitments are computationally binding if the Discrete Logarithm Problem is hard in $\mathbb{G}$ then this implies that $b_i = b_i^2$ that is $b_i \in \{0,1\}$. To do so, we observe that it is enough to prove the existence of $(b_i, r_i, s_i) \in \mathbb{F}_q$ such that

$$\begin{bmatrix} k & h & 1 \\ c_i & 1 & h \end{bmatrix} \cdot \begin{pmatrix} b_i \\ r_i \\ s_i \end{pmatrix} = \begin{bmatrix} c_i \\ c_i \end{bmatrix} \iff \begin{cases} c_i = k^{b_i} h^{r_i} \\ c_i = c_i^{b_i} h^{s_i} \end{cases} \iff \begin{cases} c_i = k^{b_i} h^{r_i} \\ c_i = k^{b_i^2} h^{b_i r_i + s_i} \end{cases}$$

where $s_i = (1 - b_i) r_i$. Again, this can be proven with a sigma protocol for $\mathcal{R}_{\mathsf{Lin}}$. Finally the knowledge of ρ is tested with a standard Schnorr proof. In conclusion, calling $\mathsf{SP.P}_{\mathsf{Lin}}, \mathsf{SP.V}_{\mathsf{Lin}}, \mathsf{SP.S}_{\mathsf{Lin}}$ respectively the prover, verifier and simulator for the sigma protocol that proves $\mathcal{R}_{\mathsf{Lin}}$, we provide a formal description of the protocol as follows

Proposition 3. *If the Discrete Logarithm Problem is hard in $\mathbb{G}$ then Protocol 8 is a sigma protocol for $\mathcal{R}_{\mathsf{Rng}}$.*

$\mathcal{P}_{\mathsf{Rng}}(g,h,k,u,v,[2^\mu],r,m):$		$\mathcal{V}_{\mathsf{Rng}}(g,h,k,u,v,[2^\mu]):$
$r_i \leftarrow^{\$} \mathbb{F}_q,\ m = \sum_{i=0}^{\mu-1} b_i 2^i$		
$s_i \leftarrow (1-b_i)r_i,\ c_i \leftarrow h^{r_i} k^{b_i}$		
$\rho \leftarrow r - \sum_{i=0}^{\mu-1} r_i 2^i$		
$c_\mu \leftarrow c \cdot \prod_{i=0}^{\mu-1} c_i^{-2^i}$		
$A \leftarrow ((g,1),(h,k))$		
$A_i \leftarrow ((k,h,1),(c_i,1,h))$		
$B \leftarrow (u,v),\ B_i \leftarrow (c_i,c_i)$		
$\mathbf{v}_i = (b_i,r_i,s_i)$		
$C \leftarrow^{\$} \mathsf{SP.P}_{\mathsf{Lin}}(A,B,(r,m))$		
$C_i \leftarrow^{\$} \mathsf{SP.P}_{\mathsf{Lin}}(A_i,B_i,\mathbf{v}_i)$		
$C_\mu \leftarrow^{\$} \mathsf{SP.P}_{\mathsf{Lin}}(h,c_\mu,\rho)$		
	$\xrightarrow{C, C_\mu, \{c_i, C_i\}_{i=0}^{\mu-1}}$	
	$\xleftarrow{e}$	$e \leftarrow^{\$} \mathbb{F}_q$
$Z \leftarrow \mathsf{SP.P}_{\mathsf{Lin}}(A,B,(r,m),C,e)$		
$Z_i \leftarrow \mathsf{SP.P}_{\mathsf{Lin}}(A_i,B_i,\mathbf{v}_i,C_i,e)$		
$Z_\mu \leftarrow \mathsf{SP.P}_{\mathsf{Lin}}(h,c_\mu,\rho,C_\mu,e)$		
	$\xrightarrow{Z, Z_\mu, \{Z_i\}_{i=0}^{\mu-1}}$	
		$A \leftarrow ((g,1),(h,k))$
		$A_i \leftarrow ((k,h,1),(c_i,1,h))$
		$B \leftarrow (u,v),\ B_i \leftarrow (c_i,c_i)$
		$c_\mu \leftarrow c \cdot \prod_{i=0}^{\mu-1} c_i^{-2^i}$
		$\beta \leftarrow \mathsf{SP.V}_{\mathsf{Lin}}(A,B,C,e,Z)$
		$\beta_i \leftarrow \mathsf{SP.V}_{\mathsf{Lin}}(A_i,B_i,C_i,e,Z_i)$
		$\beta_\mu \leftarrow \mathsf{SP.V}_{\mathsf{Lin}}(h,c_\mu,C_\mu,e,Z_\mu)$
		Return $\beta \wedge \beta_0 \wedge \ldots \wedge \beta_\mu$

Fig. 8: Sigma protocol for the $\mathcal{R}_{\mathsf{Rng}}$ relation

C Comparing UC and Game-Based Definition

C.1 Proof of Proposition 2

Proof. Let $\mathcal{S}$ be a simulator such that Π is indistinguishable from $\mathcal{F}_{\mathsf{SSLE}} \circ \mathcal{S}$ for the class of efficient environments $\mathcal{Z}$ introduced in definition 9. In the following we first argue that the composition of any PPTadversary $\mathcal{A}$ with $\mathcal{C}$ the challenger of Uniqueness, Fairness or Unpredictability Experiment respectively is an environment for Π in the above class. Then we show the three properties hold replacing Π with $\mathcal{F}_{\mathsf{SSLE}} \circ \mathcal{S}$.

Uniqueness: First of all for any efficient $\mathcal{A}$ corrupting $|M| < t$, observe that $\mathcal{A} \circ \mathcal{C}$ only requests an election to $\mathcal{F}_{\mathsf{SSLE}}$ when $|R \cap M| < \vartheta(|R|)$. Therefore if P_k is an honest user, when it returns (won, eid, R) or (lost, eid, R) we have $|R \cap M| < \vartheta(|R|)$.
Next we claim that, calling E the set used by $\mathcal{F}_{\mathsf{SSLE}}$ to record past elections (see Section 3), $(eid, i, R), (eid, j, R) \in E$ implies $i = j$. Assume by contradiction that $i \neq j$ and suppose (eid, i, R) is added first. Then (eid, j, R) would not be added afterwards as $\mathcal{F}_{\mathsf{SSLE}}$ ignores any election request with the same *eid*.
To prove uniqueness suppose by contradiction that there exists an election ID *eid* and two distinct players P_i, P_j such that $\mathsf{SSLE.Vrf}_{\mathsf{pp}}(eid, \pi, i) = \mathsf{SSLE.Vrf}_{\mathsf{pp}}(eid, \pi, j) = 1$ and let $k \in [N] \setminus M$. As a consequence P_k receives from $\mathcal{F}_{\mathsf{SSLE}}$ both $(\mathsf{result}, eid, i)$ and $(\mathsf{result}, eid, j)$. By construction this implies $(eid, i, R), (eid, j, R) \in E$ and in particular $i = j$.

Fairness: For any efficient $\mathcal{A}$ corrupting $|M| < t$ users, as before the composition $\mathcal{A} \circ \mathcal{C}$, with the latter being the challenger of the Fairness experiment, only requests election when $|R \cap M| < \vartheta(|R|)$. Next let R be the set of registered users when $\mathcal{A}$ sends chall, $n = |R|$ and $\tau = |R \cap M|$. By construction during the last election $\mathcal{F}_{\mathsf{SSLE}}$ samples a random $j \xleftarrow{\$} R$ and sends (won, eid) to P_j. Also by construction this the only player that by sending (reveal, eid) can make $\mathcal{F}_{\mathsf{SSLE}}$ broadcasts $(\mathsf{result}, eid, j)$. In particular $\mathsf{SSLE.Vrf}_{\mathsf{pp}}(eid, \perp, i) = 1$ holds only when $j = i$. It follows that

$$\Pr\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{Fair}}(1^\lambda, N, \vartheta) = 1\right] \;=\; \Pr\left[j \in R \setminus M\right] \;=\; \frac{|R \setminus M|}{|R|} \;=\; \frac{n-\tau}{n}.$$

Unpredictability: For any efficient $\mathcal{A}$ corrupting $|M| < t$ users, as before the composition $\mathcal{A} \circ \mathcal{C}$, with the latter being the challenger of the Unpredictability experiment, only requests election when $|R \cap M| < \vartheta(|R|)$.
Let R be the set of registered users when $\mathcal{A}$ sends chall. In this phase, calling (eid, j, R) the last tuple added to E in this phase by $\mathcal{F}_{\mathsf{SSLE}}$, then j is independent from all the previous messages sent by $\mathcal{F}_{\mathsf{SSLE}}$. Moreover conditioning on HW, that is equivalent to $j \in R \setminus M$, and calling j' the last message sent by $\mathcal{A}$

$$\Pr\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{Unpr}}(1^\lambda, N, \vartheta) = 1\right] \;=\; \Pr\left[j' = j | j \in R \setminus M\right] \;=\; \frac{1}{|R|}\frac{|R|}{|R \setminus M|} \;=\; \frac{1}{n-\tau}.$$

C.2 SSLE schemes from Static and Parametrised SSLE

In this section we show how to a adapt a protocol Π realising $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ to an SSLE scheme with relaxed security guarantees. Since we will need to interact with several instances of the protocol we now explicitly write the session ID. For simplicity we say "Run (Π_n, sid) with players in R" to mean that ordering $R = \{r_1, \dots, r_n\}$, P_{r_i} will interact with Π_n and session ID sid with player index i. Conversely ordering $[N] \setminus R = \{s_1, \dots, s_{N-n}\}$, P_{s_i} interacts with Π_n and session ID sid as the dummy player $\bar{P}_i$. In particular they can only receive result or rejected messages.

```
SSLE.Setup:
  $P_i$ sets $(\mathsf{pp}, \mathsf{sp}_i, \mathsf{cnt}) \leftarrow (\perp, \perp, 0)$
    $sid \leftarrow \varnothing$
    $R \leftarrow \varnothing$
    $n \leftarrow 0$

SSLE.Elect_pp:
  All players send $(\mathsf{elect}, sid, \mathsf{cnt})$ to $\Pi_n$
  $\mathsf{cnt} \leftarrow \mathsf{cnt} + 1$
  When $\Pi_n$ sends $(\mathsf{won}, sid, eid)$
    Return $(n, sid, eid)$
  When $\Pi_n$ send $(\mathsf{lost}, sid, eid)$
    Return $(n, sid, eid)$

SSLE.Reg_pp(i):
  $R \leftarrow R \cup \{i\}, \quad n \leftarrow |R|$
  $sid \leftarrow R$
  Run $(\Pi_n, sid)$ with players in $R$

SSLE.Claim_pp(c, sp_i, i):
  Parse $c = (n', sid', eid')$
  Send $(\mathsf{reveal}, sid', eid')$ to $\Pi_{n'}$
  Return $\pi \leftarrow \perp$

SSLE.Rev_pp(i):
  $R \leftarrow R \setminus \{i\}, \quad n \leftarrow |R|$
  $sid \leftarrow R$
  Run $(\Pi_n, sid)$ with players in $R$

SSLE.Vrf_pp(c, π, i):
  Parse $c = (n', sid', eid')$; When:
    $(\mathsf{result}, sid', eid', i) \leftarrow \Pi_{n'}$ return 1
    $(\mathsf{rejected}, sid', eid', i) \leftarrow \Pi_{n'}$ return 0
```

Fig. 9: The derived SSLE from Π realising $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$

Proposition 4. *Given a family of protocols Π_n that securely realise $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ among n players and $N-n$ dummy players for the class of environments that statically corrupts $T < \vartheta(n)$ players then the derived SSLE scheme defined in Figure 9 satisfy ϑ-threshold uniqueness, ϑ-threshold $2^{-\kappa}$-fairness and ϑ-threshold $\eta(\kappa)$-unpredictability with*

$$\eta(\kappa) \; = \; \max_{n \in [N]+1} \left(\frac{n}{n - \vartheta(n)} \right) \frac{1}{2^{\kappa}}.$$

Proof. Let $\mathcal{S}_n$ be a simulator such that Π_n is indistinguishable from $\mathcal{S}_n \circ \mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ for any PPTenvironment corrupting less than $\vartheta(n)$ users. We show that the composition of any adversary $\mathcal{A}$ that corrupts players in M and the challenger respectively of the Uniqueness, Fairness or Unpredictability experiment with parameters $(1^{\lambda}, N, \vartheta)$ lies in this class of environments and then prove that the property hold for $\mathcal{S} \circ \mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$.

Uniqueness: First of all let $\mathcal{C}$ be the challenger that instantiate the Uniqueness Experiment. For any $\mathcal{A}$, the composition $\mathcal{A} \circ \mathcal{C}$ is a PPTalgorithm that only performs elections with Π_n when $|R \cap M| < \vartheta(n)$ – therefore only when less than $\vartheta(n)$ non-dummy users are malicious. Therefore for $\mathcal{A} \circ \mathcal{C}$, Π_n is indistinguishable from $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}} \circ \mathcal{S}_n$.
Next we shall show that, calling E the set used by $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ to register past elections – see Section 3.1 – $(eid, i), (eid, j) \in E$ implies $i = j$. By contradiction assume that $i \neq j$ and without loss of

generality that (eid, i) is added first. Then, when $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ is invoked again to register (eid, j), eid was already used so the functionality ignores the request.
By contradiction now assume that for a given $sid = R$ there exist two different players P_i, P_j such that $\mathsf{SSLE.Vrf}(c, \pi, i) = \mathsf{SSLE.Vrf}(c, \pi, j) = 1$. If $k \in [N] \setminus M$ is the index of an uncorrupted player then P_k receives from $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ two messages $(\mathsf{reveal}, sid, eid, i)$ and $(\mathsf{reveal}, sid, eid, j)$. By construction this implies $(eid, i), (eid, j) \in E$ that is $i = j$.

Fairness: As before, calling $\mathcal{C}$ the challenger of the Fairness experiment, for any $\mathcal{A}$ the composition $\mathcal{A} \circ \mathcal{C}$ only interacts with Π_n when $|R \cap M| < \vartheta(n)$. Therefore for $\mathcal{A} \circ \mathcal{C}$, Π_n is indistinguishable from $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}} \circ \mathcal{S}_n$.
Next, call R with $n = |R|$ and $\tau = |R \cap M|$ the set of registered users when $\mathcal{A}$ send chall and $\tau < \vartheta(n)$. Without loss of generality we assume $R = [n]$. Let bad be the event that $\mathcal{F}^{k}_{\mathsf{SSSLE}}$ sends $(\mathsf{corrupted}, R, eid)$ to $\mathcal{S}_n$. By construction $\Pr[\mathsf{bad}] \leq 2^{-\kappa}$. Let also (eid, j) be the last value $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ adds to his list after this election.
If $\neg\mathsf{bad}$ then $j \sim U([n])$ by definition and P_j is the only winner, i.e. the only one for which $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ broadcasts $(\mathsf{result}, R, eid, j)$. Therefore

$$\Pr\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{Fair}}(1^\lambda, N, \vartheta) = 1 | \neg\mathsf{bad}\right] \;=\; \Pr\left[j \in [n] \setminus M | \neg\mathsf{bad}\right] \;=\; \frac{n-\tau}{n}.$$

We can so conclude that $\left|\Pr\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{Fair}}(1^\lambda, N, \vartheta) = 1\right] - \frac{n-\tau}{n}\right| \;=$

$$\left|\Pr\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{Fair}}(1^\lambda, N, \vartheta) = 1 | \mathsf{bad}\right] - \frac{n-\tau}{n}\right| \Pr[\mathsf{bad}] \;\leq\; \max\left(\frac{\tau}{n}, \frac{n-\tau}{n}\right)\frac{1}{2^\kappa} \;\leq\; \frac{1}{2^\kappa}.$$

Unpredictability: As before if $\mathcal{C}$ is the challenger of the unpredictability experiment, $\mathcal{A} \circ \mathcal{C}$ invoke Π_n only when less that $\vartheta(n)$ users are corrupted.
Also as before let R with $n = |R|$ and $\tau = |R \cap M|$ be the set of registered users when $\mathcal{A}$ send chall and $\tau < \vartheta(n)$ (wlog $R = [n]$), $H = R \setminus M$ the set of honest registered users, bad the event that $\mathcal{F}^{k}_{\mathsf{SSSLE}}$ sends $(\mathsf{corrupted}, R, eid)$ to $\mathcal{S}_n$ and (eid, j) the last message $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ store in his list. Clearly there is a honest winner if and only if $j \in [n] \setminus M$. Calling j' the output of $\mathcal{A}$ we have that

$$\Pr\left[j = j' | j \in H,\ \neg\mathsf{bad}\right] \;\leq\; \frac{\Pr\left[j = j' | \neg\mathsf{bad}\right]}{\Pr\left[j \in H | \neg\mathsf{bad}\right]} \;=\; \frac{1}{n} \cdot \frac{n}{n-\tau} \;=\; \frac{1}{n-\tau}.$$

Therefore the probability that the experiment succeed is

$$\begin{aligned}\Pr\left[\mathsf{Exp}^{\mathcal{A}}_{\mathsf{Unpr}}(1^\lambda, N, \vartheta) = 1\right] \;&=\; \Pr\left[j = j' | j \in H,\ \neg\mathsf{bad}\right]\Pr\left[\neg\mathsf{bad} | j \in H\right] + \\ &\quad + \Pr\left[j = j' | j \in H,\ \mathsf{bad}\right]\Pr\left[\mathsf{bad} | j \in H\right] \;\leq \\ &\leq \frac{1}{n-\tau} \cdot \Pr\left[\neg\mathsf{bad} | j \in H\right] + \Pr\left[\mathsf{bad} | j \in H\right] \;= \\ &= \frac{1}{n-\tau} + \frac{n-\tau-1}{n-\tau} \cdot \Pr\left[\mathsf{bad} | j \in H\right] \;\leq \\ &\leq \frac{1}{n-\tau} + \Pr\left[\mathsf{bad} | j \in H\right].\end{aligned}$$

To evaluate the last term we set $p_1 = \Pr[j \in H | \mathsf{bad}]$ and $p_2 = \Pr[\mathsf{bad}]$. Then

$$\begin{aligned}
\Pr[\mathsf{bad} | j \in H] &= \frac{\Pr[j \in H|\mathsf{bad}]\Pr[\mathsf{bad}]}{\Pr[j \in H]} = \\
&= \frac{\Pr[j \in H|\mathsf{bad}]\Pr[\mathsf{bad}]}{\Pr[j \in H|\mathsf{bad}]\Pr[\mathsf{bad}] + \Pr[j \in H|\neg\mathsf{bad}]\Pr[\neg\mathsf{bad}]} = \\
&= p_1 p_2 \left(p_1 p_2 + \frac{n-\tau}{n}(1-p_2) \right)^{-1} \leq \\
&\leq \frac{1}{2^\kappa}\left(\frac{1}{2^\kappa} + \frac{n-\tau}{n}(1-2^{-\kappa}) \right)^{-1} = \\
&= \frac{n}{n + (n-\tau)(2^\kappa - 1)} \leq \\
&\leq \frac{n}{n-\tau}\frac{1}{2^\kappa} \leq \frac{n}{n-\vartheta(n)}\frac{1}{2^\kappa} \leq \max_{n \in [N]+1}\left(\frac{n}{n-\vartheta(n)} \right)\frac{1}{2^\kappa}
\end{aligned}$$

where the first inequality follows from standard calculus techniques recalling that $(p_1, p_2) \in [0,1] \times [0, 2^{-\kappa}]$.

C.3 UC-SSLE from Static and Parametrised SSLE

In this section we show how to realise $\mathcal{F}_{\mathsf{SSLE}}$ in the $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$-hybrid model when $\kappa = \Theta(\lambda)$. Although the result and construction presented here are related to those in Section C.2 they are ultimately incomparable due to the fact that now κ has the same size of the security parameter. As in previous section the idea is to run several instances of $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ with different session ID associated to the group of currently registered users.

Proposition 5. *Protocol* $\{P^i_{\mathsf{SSLE}} : i \in [N]\}$ *realises* $\mathcal{F}_{\mathsf{SSLE}}$ *in the* $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$*-hybrid model if* $\kappa = \Theta(\lambda)$.

Proof. Below we provide a trivial simulator $\mathcal{S}$ such that $\mathcal{F}_{\mathsf{SSLE}} \circ \mathcal{S}$ is identical to the real protocol up to negligible probability.
Description of $\mathcal{S}$: Initially wait for $\mathcal{Z}$ to send $M \subseteq [N]$ the set of corrupted parties. Upon receiving:

– A request to send $(\mathsf{registered}, i)$: broadcast $(\mathsf{register\_request})$ to malicious users as P_i. Let $\mathcal{F}_{\mathsf{SSLE}}$ send his message.

– $(\mathsf{register\_request})$ from P_j: send $(\mathsf{register})$ as P_j to $\mathcal{F}_{\mathsf{SSLE}}$

– A request to send $(\mathsf{revoked}, i)$: broadcast $(\mathsf{revoke\_request})$ to malicious users as P_i. Let $\mathcal{F}_{\mathsf{SSLE}}$ send his message.

– $(\mathsf{revoke\_request})$ from P_j: send $(\mathsf{register})$ as P_j to $\mathcal{F}_{\mathsf{SSLE}}$

– A request to send (won, eid, R), (lost, eid, R) messages: send $(\mathsf{electing}, R, eid)$ to $\mathcal{Z}$:
Upon receiving $(\mathsf{prob}, R, eid, p)$: For $j \in M \cap R$, if (won, eid, R) was directed to P_j send (won, R, eid) to P_j. If (lost, eid, R) was directed to P_j send (lost, R, eid). Let $\mathcal{F}_{\mathsf{SSLE}}$ send his messages

Party $P^{(i)}_{\mathsf{SSLE}}$ realising $\mathcal{F}_{\mathsf{SSLE}}$:

Initially set $C, R \leftarrow \varnothing$:

- $(\mathsf{register})$: Broadcast $(\mathsf{register\_request})$
- $(\mathsf{register\_request})$ from P_j: Add $R \leftarrow R \cup \{j\}$; Return $(\mathsf{registered}, i)$
- (revoke): Broadcast $\mathsf{revoke\_request}$
- $(\mathsf{revoke\_request})$ from P_j: Remove $R \leftarrow R \setminus \{j\}$; Return $(\mathsf{revoked}, i)$
- (elect, eid) with $(eid, \cdot) \notin C$: order $R = \{r_1, \dots, r_n\}$ and $[N] \setminus R = \{s_1, \dots, s_{N-n}\}$. If $i = r_k$ send (elect, R, eid) to $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ as P_k. If $i = s_k$ execute $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ as $\bar{P}_k$. Add $C \leftarrow C \cup \{(eid, R)\}$
- (won, sid, eid) from $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$: Return (won, eid, sid)
- $(\mathsf{lost}, sid, eid)$ from $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$: Return $(\mathsf{lost}, eid, sid)$
- (reveal, eid): if $(eid, R') \in C$, $R' = \{r_1, \dots, r_n\}$ and $i = r_k$ send (reveal, eid) to $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ as P_k
- $(\mathsf{result}, sid, eid, j)$ from $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$: return $(\mathsf{result}, eid, sid, j)$
- $(\mathsf{rejected}, sid, eid, j)$ from $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$: return $(\mathsf{rejected}, eid, sid, j)$

- A request to send $(\mathsf{result}, eid, R, j)$: Broadcast $(\mathsf{result}, R, eid, j)$ to malicious players. Let $\mathcal{F}_{\mathsf{SSLE}}$ send his message
- A request to send $(\mathsf{rejected}, eid, R, j)$: Broadcast $(\mathsf{rejected}, R, eid, j)$ to malicious players. Let $\mathcal{F}_{\mathsf{SSLE}}$ send his message
- (reveal, eid) from P_j: Send (reveal, eid) to $\mathcal{F}_{\mathsf{SSLE}}$
- $(\mathsf{fake\_rejected}, R, eid, j)$ from $\mathcal{Z}$: send $(\mathsf{fake\_rejected}, eid, j)$ to $\mathcal{F}_{\mathsf{SSLE}}$

It is easy to observe that the real protocol differ from $\mathcal{F}_{\mathsf{SSLE}} \circ \mathcal{S}$ only when some instance of $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ returns $(\mathsf{corrupted}, R, eid)$ in the real execution. However, if we call L and upper bound on the number of elections requested by $\mathcal{Z}$ then with a union bound the probability that in some election $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ returns $(\mathsf{corrupted}, R, eid)$ is smaller than $L \cdot 2^{-\kappa}$ that is negligible when $\kappa = \Theta(\lambda)$.

D Postponed Proofs

D.1 Selective Security of FE for Orthogonality

Proof (of Proposition 1). We proceed with a sequence of hybrid games as in [Wee17]. Let $\mathbf{x}_0$ and $\mathbf{x}_1$ be the message returned by $\mathcal{A}(1^\lambda)$. Then define

– H_0^b: The selective security game where the challenger encrypts $\mathbf{x}_b$.

– H_1^b: Given $\mathbf{k} \sim U(\mathbb{F}_q)$, the challenge ciphertext is replaced by

$$c^{(1)} = \Big([\mathbf{k}]_1, \Big[\mathbf{k}^\top(x_{b,1}\mathbf{u} + \mathbf{w}_1)\Big]_1, \ldots, \Big[\mathbf{k}^\top(x_{b,n}\mathbf{u} + \mathbf{w}_n)\Big]_1\Big)$$

– H_2^b: As H_1^b but the challenger aborts if $\mathbf{a}, \mathbf{k}$ are proportional. Otherwise he computes $\widehat{\mathbf{a}}$ such that $\mathbf{a}^\top\widehat{\mathbf{a}} = 0$ and $\mathbf{k}^\top\widehat{\mathbf{a}} = 1$ and set for $\mathbf{w}_i^* \sim U(\mathbb{F}_q^2)$

$$\begin{aligned}
\mathsf{mpk}^{(2)} &= \Big([\mathbf{a}]_1, \Big[\mathbf{a}^\top\mathbf{u}\Big]_1, \Big[\mathbf{a}^\top\mathbf{w}_1^*\Big]_1, \ldots, \Big[\mathbf{a}^\top\mathbf{w}_n^*\Big]_1\Big) \\
c^{(2)} &= \Big([\mathbf{k}]_1, \Big[\mathbf{k}^\top\mathbf{w}_1^*\Big]_1, \ldots, \Big[\mathbf{k}^\top\mathbf{w}_n^*\Big]_1\Big) \\
\mathsf{sk}_{\mathbf{y}}^{(2)} &= \Big[\sum\nolimits_{i=1}^n r y_i \mathbf{w}_i - r(\mathbf{k}^\top\mathbf{u})(\mathbf{x}_b^\top\mathbf{y}) \cdot \widehat{\mathbf{a}}\Big]_2, [r]_2
\end{aligned}$$

– H_3^b: As H_2^b but $\mathsf{sk}_{\mathbf{y}}$ is generated by sampling a fresh $\delta \xleftarrow{\$} \mathbb{F}_q$ and returning

$$\mathsf{sk}_{\mathbf{y}}^{(3)} = \Big[\sum\nolimits_{i=1}^n r y_i \mathbf{w}_i - \delta(\mathbf{x}_b^\top\mathbf{y})\widehat{\mathbf{a}}\Big]_2, [r]_2$$

The thesis follows if H_0^0 is indistinguishable from H_0^1. To this aim we argue that $\mathsf{H}_0^b, \mathsf{H}_1^b$ cannot be distinguished if DDH is hard in $\mathbb{G}_1$, $\mathsf{H}_1^b, \mathsf{H}_2^b$ are statistically close, $\mathsf{H}_2^b, \mathsf{H}_3^b$ are indistinguishable assuming DDH is hard over $\mathbb{G}_2$ and $\mathsf{H}_2^0, \mathsf{H}_2^1$ are equally distributed.

– $\mathsf{H}_0^b - \mathsf{H}_1^b$. For any distinguisher $\mathcal{D}$ we define $\mathcal{A}$ breaking DDH over $\mathbb{G}_1$.

Adversary $\mathcal{A}(1^\lambda, [\alpha]_1, [\beta]_1, [\gamma]_1)$ breaking DDH over $\mathbb{G}_1$

```
1 : Sample $\rho \xleftarrow{\$} \mathbb{F}_q$ and set $[\mathbf{a}]_1 \leftarrow [(\rho, \rho\alpha)]_1$, $[\mathbf{k}]_1 \leftarrow [(\beta, \gamma)]_1$
2 : Sample $\mathbf{u}, \mathbf{w}_1, \ldots, \mathbf{w}_n \xleftarrow{\$} \mathbb{F}_q^2$ with $\mathbf{a}^\top\mathbf{u} \neq 0$ and run $(\mathbf{x}_0, \mathbf{x}_1) \xleftarrow{\$} \mathcal{D}(1^\lambda)$
3 : Compute $\mathsf{mpk} \leftarrow ([\mathbf{a}]_1, [\mathbf{a}^\top\mathbf{u}]_1, [\mathbf{a}^\top\mathbf{w}_1]_1, \ldots, [\mathbf{a}^\top\mathbf{w}_n]_1)$
4 : Set $c \leftarrow ([\mathbf{k}]_1, [\mathbf{k}^\top(x_{b,1}\mathbf{u} + \mathbf{w}_1)]_1, \ldots, [\mathbf{k}^\top(x_{b,n}\mathbf{u} + \mathbf{w}_n)]_1)$
5 : Send $\mathcal{D} \leftarrow \mathsf{mpk}, c$
6 : When $\mathbf{y} \leftarrow \mathcal{D}$: If only one of $\mathbf{x}_0^\top\mathbf{y}, \mathbf{x}_1^\top\mathbf{y}$ is zero send $\mathcal{D} \leftarrow \perp$. Else:
7 :     Compute $\mathsf{sk}_{\mathbf{y}} \leftarrow [\sum_{i=1}^n r y_i \mathbf{w}_i]_2, [r]_2$ and send $\mathcal{D} \leftarrow \mathsf{sk}_{\mathbf{y}}$
8 : When $b' \leftarrow \mathcal{D}$: Return $b'$.
```

Recall than in DDH^1 calling $s = \beta\rho^{-1} \sim U(\mathbb{F}_q)$ then $\mathbf{k} = s\mathbf{a}$ with $s \neq 0$ up to negligible probability, while $\mathbf{a}, \mathbf{k}$ are uniform and independent in DDH^0. Therefore in the first case $\mathcal{A}$ perfectly simulates H_0^b, while in the second H_1^b. As a consequence $\mathcal{D}$ has the same advantage of $\mathcal{A}$ that is negligible.

– $H_1^b - H_2^b$. Observe that in H_1^b, $\mathbf{a}, \mathbf{k}$ are uniform over $\mathbb{F}_q^2$, hence the probability that $\mathbf{k}$ lies in the linear span of $\mathbf{a}$ is q^{-1}. Up to this negligible probability $\mathbf{a}, \mathbf{k}$ are independent, so there exists a unique vector $\widehat{\mathbf{a}}$ satisfying $\mathbf{a}^\top \widehat{\mathbf{a}} = 0$ and $\mathbf{k}^\top \widehat{\mathbf{a}} = 1$. Calling $\mathbf{w}_i^* = \mathbf{w}_i + x_{i,b}(\mathbf{k}^\top \mathbf{u}) \cdot \widehat{\mathbf{a}}$ this is still uniformly distributed and we obtain the distribution of game H_2^b since

$$\begin{aligned} \mathbf{a}^\top \mathbf{w}_i &= \mathbf{a}^\top(\mathbf{w}_i^* - x_i(\mathbf{k}^\top \mathbf{u}) \cdot \widehat{\mathbf{a}}) = \mathbf{a}^\top \mathbf{w}_i^* \\ \mathbf{k}^\top(x_{i,b}\mathbf{u} + \mathbf{w}_i) &= x_{i,b}(\mathbf{k}^\top \mathbf{u}) + \mathbf{k}^\top(\mathbf{w}_i^* - x_{i,b}(\mathbf{k}^\top \mathbf{u}) \cdot \widehat{\mathbf{a}}) = \mathbf{k}^\top \mathbf{w}_i^* \\ \sum_{i=1}^n y_i \mathbf{w}_i &= \sum_{i=1}^n y_i(\mathbf{w}_i^* - x_{i,b}(\mathbf{k}^\top \mathbf{u})) \cdot \widehat{\mathbf{a}} = \sum_{i=1}^n y_i \mathbf{w}_i^* - (\mathbf{x}_b^\top \mathbf{y})(\mathbf{k}^\top \mathbf{u}) \cdot \widehat{\mathbf{a}}. \end{aligned}$$

Hence the statistical distance from this two games is smaller than q^{-1}.

– $\mathsf{H}_2^b - \mathsf{H}_3^b$. For any distinguisher $\mathcal{D}$ that query at most ℓ keys, we define $\mathcal{A}$ that plays against DDH_ℓ over $\mathbb{G}_2$.

Adversary $\mathcal{A}(1^\lambda, [r_1]_2, \dots, [r_\ell]_2, [\sigma]_2, [\delta_1]_2, \dots, [\delta_\ell]_2)$ for DDH_ℓ over $\mathbb{G}_2$

```
1 : Sample $\mathbf{a}, \mathbf{k}, \mathbf{w}_1, \dots, \mathbf{w}_n \leftarrow^{\$} \mathbb{F}_q^2$, $u \leftarrow^{\$} \mathbb{F}_q$ and run $\mathcal{D}(1^\lambda) \to (\mathbf{x}_0, \mathbf{x}_1)$
2 : Compute $\widehat{\mathbf{a}}$ such that $\mathbf{k}^\top \widehat{\mathbf{a}} = 1$ and $\mathbf{a}^\top \widehat{\mathbf{a}} = 0$
3 : Set $\mathsf{mpk} \leftarrow ([\mathbf{a}]_1, [u]_1, [\mathbf{a}^\top \mathbf{w}_1]_1, \dots, [\mathbf{a}^\top \mathbf{w}_n]_1)$
4 : Set $c \leftarrow ([\mathbf{k}]_1, [\mathbf{k}^\top \mathbf{w}_1]_1, \dots, [\mathbf{k}^\top \mathbf{w}_n]_1)$ and send $\mathcal{D} \leftarrow \mathsf{mpk}, c$
5 : The $j$-th time $\mathcal{D} \to \mathbf{y}$: if only one of $\mathbf{x}_0^\top \mathbf{y}$, $\mathbf{x}_1^\top \mathbf{y}$ is zero send $\mathcal{D} \leftarrow \perp$. Else:
6 :     Set $\mathsf{sk}_\mathbf{y} \leftarrow [\sum_{i=1}^n r_j y_i \mathbf{w}_i - \delta_j(\mathbf{x}_b^\top \mathbf{y}) \cdot \widehat{\mathbf{a}}]_2, [r_j]_2$ and send $\mathcal{D} \leftarrow \mathsf{sk}_\mathbf{y}$
7 : When $\mathcal{D} \to b'$, return $b'$.
```

Again up to negligible probability $\mathbf{a}, \mathbf{k}$ are linearly independent and $\widehat{\mathbf{a}}$ can be computed. In this case, when $\mathcal{A}$ receives a truly random tuple in DDH_ℓ^0 it perfectly simulates H_3^b. Conversely in DDH_ℓ^1 $\delta_j = \sigma r_j$. Since in H_2^b, $\mathbf{u}$ is uniform and $\mathbf{a}, \mathbf{k}$ are linearly independent, the values $\mathbf{a}^\top \mathbf{u}$ and $\mathbf{k}^\top \mathbf{u}$ are uniformly and independently distributed. Hence, replacing $u = \mathbf{a}^\top \mathbf{u}$ and $\sigma = \mathbf{k}^\top \mathbf{u}$ we deduce that $\mathcal{A}$ in this case perfectly simulates H_2^b. It follows that $\mathcal{D}$'s advantage is negligible as it is smaller that $\mathcal{A}$'s advantage plus a negligible term.

– $\mathsf{H}_3^0 - \mathsf{H}_3^1$. In this case observe that the only difference lies in the secret key associated to $\mathbf{y}$. However, if $\mathbf{x}_0^\top \mathbf{y} = 0 = \mathbf{x}_1^\top \mathbf{y}$ in both worlds the key returned is $[\sum_{i=1}^n r y_i \mathbf{w}_i]_2, [r]_2$. If instead $\mathbf{x}_0^\top \mathbf{y} \neq 0 \neq \mathbf{x}_1^\top \mathbf{y}$ then the key returned in H_3^0 or H_3^1 follows the same distribution of

$$\left[\sum_{i=1}^n r y_i \mathbf{w}_i - \delta \widehat{\mathbf{a}}\right]_2, [r]_2.$$

To see this one can substitute $\delta = \delta^* \cdot (\mathbf{x}_b^\top \mathbf{y})$ which remains uniformly distributed over $\mathbb{F}_q$ as $(\mathbf{x}_b^\top \mathbf{y}) \neq 0$. Finally if $\mathbf{y}$ is orthogonal to only one of $\mathbf{x}_0, \mathbf{x}_1$, $\perp$ is returned in both experiments.

The proof is therefore concluded.

D.2 Proof of Theorem 1

Proof. To prove the statement we have to provide a simulator $\mathcal{S}$ that interacts with $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ such that for all environments $\mathcal{Z}$, $\mathcal{S} \circ \mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ is indistinguishable from the real protocol.
Description of $\mathcal{S}$: Initially generates $\mathsf{mpk}, \mathsf{msk} \xleftarrow{\$} \mathsf{KS.Setup}(1^{\lambda})$, the secret keys $\mathsf{sk}_i \xleftarrow{\$} \mathsf{KS.KeyGen}(i, \mathsf{msk})$ and wait for $M \xleftarrow{\$} \mathcal{Z}$. Call $i_0 = \min([N] \setminus M)$. Upon receiving

– (setup) from P_j: send $(\mathsf{input}, \mathsf{mpk}, \mathsf{sk}_j)$ to P_j

– $(\mathsf{electing}, eid)$ from $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$: send back $(\mathsf{prob}, eid, 0)$.

– (won, eid) to P_j from $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$: compute $c \xleftarrow{\$} \mathsf{KS.Enc}(j, \mathsf{mpk})$ and broadcast $(\mathsf{challenge}, eid, c)$. Let $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ send (lost, eid) to honest users.

– (lost, eid) to P_j for all $j \in M$ from $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$: compute $c \xleftarrow{\$} \mathsf{KS.Enc}(i_0, \mathsf{mpk})$ and broadcast $(\mathsf{challenge}, eid, c)$. Let $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ send (won, eid) and (lost, eid) to honest users.

– a request to broadcast $(\mathsf{result}, eid, i)$: Simulate $\pi \xleftarrow{\$} \mathsf{NIZK.S}_{\mathsf{Dec}}(\mathsf{mpk}, c, i)$ and broadcast $(\mathsf{claim}, eid, \pi)$. Let $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ send $(\mathsf{result}, eid, i)$.

– $(\mathsf{rejected}, eid, i)$ from $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$: Broadcast $(\mathsf{claim}, eid, \perp)$ on behalf of P_i.

– $(\mathsf{claim}, eid, \pi)$ from P_j: if $1 \leftarrow \mathsf{NIZK.V}_{\mathsf{Dec}}(\mathsf{mpk}, c, j, \pi)$ send (reveal, eid) to $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ as P_j. Otherwise send $(\mathsf{fake\_reject}, eid)$ to $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$.

We proceed to prove that $\mathcal{S}$ is the right simulator through a series of hybrid games. Let L be an upper bound on the number of election $\mathcal{Z}$ requests through the honest parties. Then we define

– $\mathsf{H}_{\mathsf{real}}$: the real protocol

– H_0: as $\mathsf{H}_{\mathsf{real}}$ but all the NIZK proofs of honest parties are simulated.

– H_1: as H_0 but when all honest parties request (elect, eid), sample $j \xleftarrow{\$} [N]$, $c \leftarrow \mathsf{KS.Enc}(j, \mathsf{mpk})$ and store $E \leftarrow E \cup \{(eid, j)\}$. Send $(\mathbf{won}, eid)$ to P_j if $j \notin M$, (lost, eid) to the other player in $[N] \setminus M$ and to users in M broadcast $(\mathsf{challenge}, eid, c)$.
Furthermore when P_i send (reveal, eid) check if $(eid, j) \in E$ and in this case, if $i = j$ broadcast $(\mathsf{result}, eid, i)$ to honest players and $(\mathsf{claim}, eid, \pi)$ with a simulated π to malicious parties. If $i \neq j$ instead broadcast $(\mathsf{rejected}, eid, i)$ to honest players and $(\mathsf{claim}, eid, \perp)$ to malicious parties.

– H_2: as H_1 but when P_j sends $(\mathsf{claim}, eid, \pi)$ each honest player return $(\mathsf{result}, eid, j)$ if π is accepted and $(eid, j) \in E$, $(\mathsf{rejected}, eid, j)$ otherwise.

– H_3^{ℓ}: As H_2 but in the first ℓ elections if $j \notin M$ then $c \leftarrow \mathsf{KS.Enc}(i_0, \mathsf{mpk})$

$\underline{\mathsf{H}_{\mathsf{real}} \equiv \mathsf{H}_0}$: From the perfect zero knowledge property of the NIZK arguments used the two games produces identically distributed transcripts.

$\underline{\mathsf{H}_0 \equiv \mathsf{H}_1}$: To prove that these two games are behaviourally identical we begin showing that $(eid, i) \in E$ if and only if $1 \leftarrow \mathsf{KS.Dec}(c, i, \mathsf{mpk}, \mathsf{sk}_i)$. Assuming $(eid, i) \in E$ then c is the encryption of i and in particular by correctness $1 \leftarrow \mathsf{KS.Dec}(c, i, \mathsf{mpk}, \mathsf{sk}_i)$. Conversely let c be the encryption of j and assume that $1 \leftarrow \mathsf{KS.Dec}(c, i, \mathsf{mpk}, \mathsf{sk}_i)$. Then again by correctness $j = i$, that is $(eid, i) \in E$.
We remark that in H_0 during reveal request, if the decryption algorithm returns 1 then the claim is always accepted by honest parties because of the perfect completeness of $(\mathsf{NIZK.P}_{\mathsf{Dec}}, \mathsf{NIZK.V}_{\mathsf{Dec}})$.

$\underline{\mathsf{H}_1 \equiv \mathsf{H}_2}$: By the soundness property we can assume, up to negligible probability, that all the statements proved through accepted NIZK arguments are true. Conditioning to this event the two games are then identical since each time $(\mathsf{claim}, eid, \pi)$ sent by P_j is accepted by honest players then, calling $(eid, c) \in C$, there exists sk such that $(\mathsf{mpk}, c, j, \mathsf{sk}) \in \mathcal{R}_{\mathsf{Dec}}$ (see Section 2.7). Therefore

$$(\mathsf{mpk}, i, \mathsf{sk}) \in \mathcal{L}_{\mathsf{key}}, \quad 1 \leftarrow \mathsf{KS.Dec}(c, i, \mathsf{mpk}, \mathsf{sk})$$

In particular if $(eid, j) \in E$ then c is an encryption of j and by the definition of $\mathcal{L}_{\mathsf{key}}$ we deduce that $j = i$. In conclusion we showed that each time π is accepted $(eid, j) \in E$ and therefore adding this internal check does not change the behaviour of the game.

$\underline{\mathsf{H}_3^{\ell} \equiv \mathsf{H}_3^{\ell-1}}$: We reduce the indistinguishability of these two games to the selective security of the underlying encryption scheme, i.e. given a distinguisher $\mathcal{D}$ we define an algorithm $\mathcal{A}$ breaking the selective security.
Description of $\mathcal{A}$: Initially run $M \xleftarrow{\$} \mathcal{D}$, set $i_0 = \min([N] \setminus M)$ and sample $i_1 \xleftarrow{\$} [N] \setminus M$. Send (i_0, i_1) to the challenger and wait for (mpk, c^*). The request the secret key sk_j associated to the keyword j for all $j \in M$. Initialise $\mathsf{ectn} \leftarrow 0$, $E, C \leftarrow \varnothing$ and run $\mathcal{D}$ until it returns

– (setup) on behalf of P_j: send $(\mathsf{input}, \mathsf{mpk}, \mathsf{sk}_j)$.

– (elect, eid) from all honest parties: set $\mathsf{ecnt} \leftarrow \mathsf{ecnt} + 1$ and sample $j' \xleftarrow{\$} [N]$. If $j' \in M$ set $c \leftarrow \mathsf{KS.Enc}(j, \mathsf{mpk})$ and $j \leftarrow j'$ else if $j' \notin M$

$$\begin{aligned} &\text{If } \mathsf{ecnt} < \ell: \quad c \xleftarrow{\$} \mathsf{KS.Enc}(i_0, \mathsf{mpk}), \quad j \leftarrow j' \\ &\text{If } \mathsf{ecnt} = \ell: \quad c \leftarrow c^*, \quad j \leftarrow i_1 \\ &\text{If } \mathsf{ecnt} > \ell: \quad c \xleftarrow{\$} \mathsf{KS.Enc}(j, \mathsf{mpk}), \quad j \leftarrow j'. \end{aligned}$$

Send (won, eid) to P_j if $j \notin M$, (lost, eid) to the other honest parties and $(\mathsf{challenge}, eid, c)$ to corrupted players. Store $E \leftarrow E \cup \{(eid, j)\}$ and $C \leftarrow C \cup \{(eid, j)\}$.

– (reveal, eid) from P_i with $(eid, j) \in E$: If $j \neq i$ broadcast $(\mathsf{rejected}, eid, i)$ to honest and $(\mathsf{claim}, eid, \perp)$ to dishonest parties. If $j = i$ find $(eid, c) \in C$, simulate $\pi \leftarrow \mathsf{NIZK.S}_{\mathsf{Dec}}(\mathsf{mpk}, c, i)$ and broadcast $(\mathsf{result}, eid, i)$ to honest and $(\mathsf{claim}, eid, \pi)$ to malicious users.

– $(\mathsf{claim}, eid, \pi)$ from P_j with $(eid, c) \in C$ and $(eid, j') \in E$: If $j = j'$ and $1 \xleftarrow{\$} \mathsf{NIZK.V}_{\mathsf{Dec}}(\mathsf{mpk}, c, j, \pi)$ broadcast to honest users $(\mathsf{result}, eid, j)$. Otherwise broadcast $(\mathsf{rejected}, eid, j)$.

– a bit b: Return b.

Proof of Claim: By inspection it's clear that $\mathcal{A}$ behaves as in H_3^ℓ or $\mathsf{H}_3^{\ell-1}$ when $\mathcal{D}$ sends setup, reveal or claim requests. The same applies to elect messages when $\mathsf{enct} \neq \ell$. When $\mathsf{ecnt} = \ell$ instead we begin observing that j is uniform. Indeed for any $\alpha \in [N]$, if $\alpha \in M$

$$\begin{aligned}\Pr[j=\alpha] &= \Pr\left[j=\alpha|j' \in M\right]\Pr\left[j' \in M\right] + \Pr\left[j=\alpha|j' \in [N]\setminus M\right]\Pr\left[j' \in M\right]\\ &= \Pr\left[j'=\alpha|j' \in M\right]\Pr\left[j' \in M\right] \;=\; \Pr\left[j'=\alpha\right] \;=\; 1/N.\end{aligned}$$

Similarly for all $\alpha \in [N]\setminus M$

$$\begin{aligned}\Pr[j=\alpha] &= \Pr\left[j=\alpha|j' \in M\right]\Pr\left[j' \in M\right] + \Pr\left[j=\alpha|j' \in [N]\setminus M\right]\Pr\left[j' \in M\right]\\ &= \Pr[i_1=\alpha]\Pr\left[j' \in [N]\setminus M\right] \;=\; \frac{1}{N-|M|}\cdot\frac{N-|M|}{N} \;=\; \frac{1}{N}.\end{aligned}$$

Finally call i_β the message chosen by the challenger. When $\beta = 1$ and $j' \notin M$ then c contains the encryption of $i_1 = j$ therefore $\mathcal{A}$ perfectly simulates $\mathsf{H}_3^{\ell-1}$. Conversely when $\beta = 0$ and $j' \notin M$ the c contains the encryption of i_0 as in H_3^ℓ. It follows that $\mathcal{D}$ has the same advantage of $\mathcal{A}$, that is negligible.

$\underline{\mathsf{H}_3^L \equiv \mathcal{S}\circ\mathcal{F}_{\mathsf{SSSLE}}^\kappa}$: follows by inspection.

D.3 Proof of Theorem 2

*Proof (**Proof of Theorem 2**).* In order to prove the statement we must provide a simulator $\mathcal{S}$ that interacts with $\mathcal{F}_{\mathsf{SSSLE}}^\kappa$ such that the protocol is indistinguishable from $\mathcal{S}\circ\mathcal{F}_{\mathsf{SSSLE}}^\kappa$ for any PPT environment $\mathcal{Z}$ that statically corrupts strictly less than t players. Intuitively we use the security of Threshold Elgamal and the UC-commitment to make the simulator alter the message contained in the ℓ-th ciphertext. If an honest party wins, this choice will be uniformly random. As this message cannot be decrypted by anyone with overwhelming probability, by selective security of the underlying FE scheme $\mathcal{Z}$ can't distinguish it from the encryption of a vector associated to a honest player. Reveal request are then handled simulating the associated proof and the message m properly.
A detailed description of $\mathcal{S}$ is provided below, where we omit to specify the behaviour of honest parties when this equals the correct one. We remark however that $\mathcal{S}$ can always simulate it as he initially generates all the public and private parameters.

Description of $\mathcal{S}$: Initially wait M from $\mathcal{Z}$ and forward it to $\mathcal{F}_{\mathsf{SSSLE}}^\kappa$. Set $B \leftarrow \varnothing$, generate $\mathsf{mpk}, \mathsf{msk} \xleftarrow{\$} \mathsf{FE.KeyGen}(1^\lambda)$, $\mathsf{sk}_{\phi(m),m} \leftarrow \mathsf{FE.KeyGen}((1,m),\mathsf{msk})$ and $\mathsf{sk}_i \leftarrow (\mathsf{sk}_{i,m})_{m\in\phi^{-1}(i)}$. Sample $f \xleftarrow{\$} \mathbb{F}_q[x]_{<t}$, $g \xleftarrow{\$} \mathbb{G}_1$ call $\sigma \leftarrow f(-1)$ and compute $h \leftarrow g^\sigma$, $k_i \leftarrow g^{f(i)}$. Set $\mathsf{pp} \leftarrow (\mathsf{mpk}, g, h, k_0, \ldots, k_{N-1})$ and $\mathsf{sp}_i \leftarrow (\mathsf{sk}_i, f(i))$. Upon receiving:

1. (setup) from P_j: sends $(\mathsf{input}, \mathsf{mpk}, \mathsf{sp}_j)$ to P_j

2. $(\mathsf{electing}, eid)$ from $\mathcal{F}_{\mathsf{SSSLE}}^\kappa$: Set p the probability that a random committee is malicious and send (prob, eid, p) to $\mathcal{F}_{\mathsf{SSSLE}}^\kappa$

3. A request to send (won, eid), (lost, eid) from $\mathcal{F}_{\mathsf{SSSLE}}^\kappa$: If $\mathcal{F}_{\mathsf{SSSLE}}^\kappa$ is sending (won, eid) to P_j, $j \in M$, then set $j^* \leftarrow j$, otherwise set $j^* \leftarrow \perp$. Sample $Q \subseteq [N]\setminus D$ such that $Q \nsubseteq M$ and broadcast

$(\mathsf{tossed}, eid, Q)$. Simulate honestly all player but P_i with $i = \min(Q \setminus M)$. For P_i: Sample $\alpha, \beta, \gamma, s_i \leftarrow^{\$} \mathbb{F}_q$ and $v_1, v_2 \leftarrow^{\$} \mathbb{G}_1$. Initialise $\mathsf{cnt} \leftarrow 1$ set

$$u_1 \leftarrow g^{\alpha},\ u_2 \leftarrow g^{\beta},\ v_3 \leftarrow g^{\gamma},\ v_4 \leftarrow h^{\gamma}$$

$$\widetilde{G}_i \leftarrow v_3,\ \widetilde{H}_i \leftarrow v_4,\ R_i \leftarrow u_1,\ \widetilde{c}_{1,i} \leftarrow u_2$$

and simulate $\pi^i_{\mathsf{Rng}}, \pi^{0,i}_{\mathsf{DDH}}$. Upon receiving

3.1. $(\mathsf{commit}, eid||0, \widetilde{G}_j, \widetilde{H}_j, R_j, \widetilde{c}_{1,j}, \pi^j_{\mathsf{Rng}}, \pi^j_{\mathsf{DDH}})$ from all P_j, $j \in Q \cap M$: For all j with accepting proofs extract m_j as the discrete logarithm of $R_j^{-\sigma}\widetilde{c}_{1,j}$ with base $\left[\mathbf{a}^{\top}\mathbf{u}\right]_1$.

3.2. $(\mathsf{open}, eid||0)$ form all P_j, $j \in Q \cap M$: Run P_i setting

$$G_i \leftarrow v_1 \cdot \prod_{j \neq i} \widetilde{G}_j^{\alpha}, \quad c_{1,i} \leftarrow v_2 \cdot \prod_{j \neq i} R_j^{\gamma\sigma}$$

$$\mathbf{c}_{0,i} \leftarrow \left[s_i \mathbf{a}^{\top}\mathbf{u}\right]_1, \quad c_{2,i} \leftarrow \left[s_i \mathbf{a}^{\top}(\mathbf{w}_2 - \mathbf{u})\right]_1$$

and simulating $\pi^{1,i}_{\mathsf{DDH}}, \pi^{2,i}_{\mathsf{DDH}}, \pi^{3,i}_{\mathsf{DDH}}$. $(\mathsf{recepit}, eid||\mathsf{cnt})$.

3.3. $(\mathsf{commit}, eid||\mathsf{cnt}, H_j, \pi^{3,j}_{\mathsf{DDH}})$, $(\mathsf{msg}, G_j, \mathbf{c}_{0,j}, c_{1,j}, c_{2,j}, \pi^{1,j}_{\mathsf{DDH}}, \pi^{2,j}_{\mathsf{DDH}})$ from P_j, $j \in Q \cap M$: If all the proofs are accepted set $m \leftarrow \prod_{j \neq i} m_j$, $J \leftarrow \prod_{j \neq i} \widetilde{G}_j$.
If $j^* = \perp$ set $H_i \leftarrow^{\$} \mathbb{G}_1$. Otherwise find $m_i \in [N]$ such that $\phi(m + m_i) = j^*$ and set

$$H_i \leftarrow \left(\prod_{j \neq i} H_j^{\beta}\right) \cdot v_2 \cdot \left[-s_i \mathbf{a}^{\top}((m + m_i)\mathbf{u} + \mathbf{w}_1)\right]_1 \cdot \left[x^{-1}\mathbf{a}^{\top}(m_i \mathbf{u} + \mathbf{w}_1)\right]_J.$$

Store $B \leftarrow B \cup \{(eid, m)\}$ and broadcast $(\mathsf{decom}, eid||\mathsf{cnt}, i, H_i, \pi^{3,i}_{\mathsf{DDH}})$.
Conversely if some proof is rejected update $\mathsf{cnt} \leftarrow \mathsf{cnt} + 1$ and restart from the previous point.

3.4. $(\mathsf{open}, eid||\mathsf{cnt})$ or error from P_j for all $j \in Q \cap M$: Instruct $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ to send (elect, eid).

3.5. (error) from P_j: Execute all parties honestly. If $Q = \varnothing$ simulate correctly $\mathcal{F}^{\kappa,D}_{\mathsf{CT}}$

4. $(\mathsf{corrupted}, eid)$ from $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$: Sample $Q \subseteq M$ and broadcast $(\mathsf{tossed}, eid, Q)$. During the protocol using σ extract m_j for all $j \in Q$.
If the election succeed before $Q = \varnothing$ set $m \leftarrow \sum_{j \in Q} m_j$, add $B \leftarrow B \cup \{(eid, m)\}$ and send $(\mathsf{infl}, eid, \phi(m))$.

5. $(\mathsf{result}, eid, j)$ from $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$: Find $(eid, m') \in B$, set $m \in [N] + m'$ such that $\phi(m) = j$, simulate π from $\mathsf{NIZK.S_{Dec}}$ and broadcast $(\mathsf{claim}, eid, \pi, m)$. Instruct $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ to send $(\mathsf{result}, eid, j)$.

6. $(\mathsf{claim}, eid, \pi, m)$ from P_j: If $\phi(m) = j$ and π is accepted, send (reveal, eid) to $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$ as P_j, otherwise send $(\mathsf{fake\_rejected}, eid, j)$ to $\mathcal{F}^{\kappa}_{\mathsf{SSSLE}}$.

Next, called L the maximum number of election that $\mathcal{Z}$ requests through honest parties, we provide a sequence of hybrid games $\mathsf{H}^0_0, \ldots, \mathsf{H}^3_\ell$ where $H^{\ell}_{\cdot}$ modifies the ℓ-th election when $Q \not\subseteq M$ by altering the messages sent by P_i with $i = \min(Q \setminus M)$. To avoid repetitions observe that messages contained in the first commitment sent in a committee, i.e. $\widetilde{G}_j, \widetilde{H}_j, R_j, \widetilde{c}_{1,j}$, uniquely identifies $s_j, r_j, m_j \in \mathbb{F}_q$ such that $\widetilde{G}_j = g^{s_j}$, $R_j = g^{r_j}$, $\widetilde{c}_{1,j} = h^{r_j}\left[m_j \mathbf{a}^{\top}\mathbf{u}\right]_1$.

– $\mathsf{H}_{\mathsf{real}}$ the real protocol.

– H_0^0 as $\mathsf{H}_{\mathsf{real}}$ but P_i simulates all the NIZK arguments.

– H_0^ℓ as $\mathsf{H}_0^{\ell-1}$, but P_i samples $u_1, u_2 \leftarrow^{\$} \mathbb{G}_1$, sets $R_i \leftarrow u_1$, $\widetilde{c}_{1,i} \leftarrow u_2 \left[m_i \mathbf{a}^\top \mathbf{u}\right]_1$, updates $s \leftarrow \sum_{j\in Q\setminus\{i\}}$ and sets

$$G_i \leftarrow u_1^s u_1^{s_i}, \quad H_i \leftarrow u_2^s u_2^{s_i}$$

– $\mathsf{H}_1^0 = \mathsf{H}_0^L$. H_1^ℓ as $\mathsf{H}_1^{\ell-1}$, but P_i samples $v_1, v_2, v_3 \leftarrow^{\$} \mathbb{G}_1$, $\theta \leftarrow^{\$} \mathbb{F}_q$ calls $v_4 \leftarrow v_3^\sigma$, where $\sigma \in \mathbb{F}_q$ such that $h = g^\sigma$, updates $s \leftarrow \sum_{j\in Q\setminus\{i\}} s_j$, $r \leftarrow \sum_{j\in Q\setminus\{i\}} r_j$, $m \leftarrow \sum_{j\in Q\setminus\{i\}}$ and sets

$$\widetilde{G}_i \leftarrow v_3, \quad \widetilde{H}_i \leftarrow v_4, \quad G_i \leftarrow u_1^s v_1, \quad H_i \leftarrow u_2^s v_2$$

$$c_{1,i} \leftarrow v_4^r v_2 \left[s_i \mathbf{a}^\top ((m+m_i)\mathbf{u} + \mathbf{w}_1)\right]_1$$

– $\mathsf{H}_2^0 = \mathsf{H}_1^L$. H_2^ℓ as $\mathsf{H}_2^{\ell-1}$, but P_i sets

$$\widetilde{c}_{1,i} \leftarrow u_2, \quad c_{1,i} \leftarrow v_4^r v_2, \quad H_i \leftarrow u_2^s v_2 \left[-s_i \mathbf{a}^\top ((m+m_i)\mathbf{u} + \mathbf{w}_1)\right]_1 \left[-s m_i \mathbf{a}^\top \mathbf{u}\right]_1$$

– $\mathsf{H}_2^\star$ as H_2^L, but initially call $B, E \leftarrow \varnothing$ and during the election with ID eid set $m \leftarrow \sum_{j\in Q\setminus\{i\}} m_j$ and add $E \leftarrow E \cup \{(eid, \phi(m+m_i))\}$, $B \leftarrow B \cup \{eid, m\}$.
For each (reveal, eid) request to P_j, if $(eid, j) \in E$ broadcast $(\mathsf{results}, eid, j)$ to honest players. Moreover given $(eid, m) \in B$, compute $m' \in [N]$ such that $\phi(m+m') = j^*$, simulate π and broadcast $(\mathsf{claim}, eid, \pi, m+m')$ to malicious parties.

– $\mathsf{H}_3^0 = \mathsf{H}_2^\star$. H_3^ℓ as $\mathsf{H}_3^{\ell-1}$ but initially set $B, E \leftarrow \varnothing$. In the ℓ-th election P_i samples a random $j^* \leftarrow^{\$} [N]$ and add $E \leftarrow E \cup \{(eid, j^*)\}$, $B \leftarrow B \cup \{(eid, m)\}$. If $j^* \in M$ sets $m_i \in [N]$ such that $\phi(m+m_i) = j^*$, otherwise sets $H_i \leftarrow^{\$} \mathbb{G}_1$.

– H_4 as H_3^L, but each time a malicious player P_j request $(\mathsf{claim}, eid, \pi, m)$, honest parties reply with $(\mathsf{result}, eid, j)$ after checking $(eid, j) \in E$, $\phi(m) = j$ and that π is accepted by $\mathsf{NIZK.V_{Dec}}$

$\underline{\mathsf{H}_{\mathsf{real}} \equiv \mathsf{H}_0^0}$: The two games follows the same distribution since the arguments used have perfect HVZK.

$\underline{\mathsf{H}_0^{\ell-1} \equiv \mathsf{H}_0^\ell}$: Given a PPT distinguisher $\mathcal{D}$, we provide an algorithm $\mathcal{A}$ that breaks DDH with almost the same advantage.
Description of $\mathcal{A}(1^\lambda, [\alpha]_1, [\beta]_1, [\gamma]_1)$: Samples $\rho \leftarrow^{\$} \mathbb{F}_q$ and computes

$$(g, h, \widehat{u}_1, \widehat{u}_2) \leftarrow ([\rho]_1, [\rho\alpha]_1, [\beta]_1, [\gamma]_1).$$

Runs $M \leftarrow^{\$} \mathcal{D}$, wlog with $|M| = t-1$. It initializes a counter $\mathsf{ecnt} \leftarrow 0$, generates $\mathsf{mpk}, \mathsf{mpk} \leftarrow^{\$} \mathsf{FE.Setup}(1^\lambda)$ and sk_i for all $i \in [N]$, samples $f_j \leftarrow^{\$} \mathbb{F}_q$ and sets $k_j \leftarrow g^{f_j}$. Calling λ_{-1}, λ_j the Lagrange coefficients to evaluate a polynomial in i from evaluations in $M \cup \{-1\}$, $\mathcal{A}$ evaluates $k_i = h^{\lambda_{-1}} \prod_{j\in M} k_j^{\lambda_j}$ for all $i \in [N] \setminus M$ and sets $\mathsf{pp} \leftarrow (\mathsf{mpk}, g, h, k_0, \ldots, k_{N-1})$, $\mathsf{sp}_j \leftarrow (\mathsf{sk}_j, f_j)$ for $j \in M$. It then executes $\mathcal{D}$ until it returns:

1. (setup) as P_j: send $(\mathsf{input}, eid, \mathsf{pp}, \mathsf{sp}_j)$ to P_j
2. (elect, eid) or (toss, eid) from all parties: update $\mathsf{ecnt} \leftarrow \mathsf{ecnt} + 1$ and simulate $\mathcal{F}_{\mathsf{CT}}^{\kappa,D}$ returning $(\mathsf{tossed}, eid, Q)$. If $\mathsf{ecnt} \leq \ell$ and $Q \nsubseteq M$ let $i = \min(Q \setminus M)$ and set

$$\begin{aligned} \text{If } \mathsf{ecnt} < \ell : &\quad u_1, u_2 \leftarrow^{\$} \mathbb{G}_1 \\ \text{If } \mathsf{ecnt} = \ell : &\quad u_1, u_2 \leftarrow \widehat{u}_1, \widehat{u}_2. \end{aligned}$$

In this case execute P_i setting $R_i \leftarrow u_1, \widetilde{c}_{1,i} \leftarrow u_2 \left[m_i \mathbf{a}^\top \mathbf{u}\right]_1$

2.1. $(\mathsf{commit}, eid||0, \widetilde{G}_j, \widetilde{H}_j, R_j, \widetilde{c}_{1,j}, \pi_{\mathsf{Rng}}^j, \pi_{\mathsf{DDH}}^{0,j})$ or error from all P_j with $j \in Q \setminus M$: extract from the accepted $\pi_{\mathsf{DDH}}^{0,j}$ and π_{Rng}^j values $s_j, r_j, m_j \in \mathbb{F}_q$ such that $\widetilde{G}_j = g^{s_j}$, $R_j = g^{r_j}$, $\widetilde{c}_{1,j} = h^{r_j} \left[m_j \mathbf{a}^\top \mathbf{u}\right]_1$.

2.2. $(\mathsf{open}, eid||0)$ or error from all P_j, $j \in M \cap Q$: If $\mathsf{ecnt} \leq \ell$ keep executing P_i simulating the proofs $\pi_{\mathsf{DDH}}^{0,i}, \pi_{\mathsf{DDH}}^{1,i}$ and setting

$$s \leftarrow \sum\nolimits_{j \in Q \setminus \{i\}} s_j, \quad G_j \leftarrow u_1^{s+s_i}, \quad H_j \leftarrow u_2^{s+s_i}.$$

2.3. $(\mathsf{msg}, eid||\mathsf{cnt}, G_j, \mathbf{c}_{0,j}, c_{1,j}, c_{2,j}, \pi_{\mathsf{DDH}}^{1,j}, \pi_{\mathsf{DDH}}^{2,j})$ and $(\mathsf{commit}, eid||\mathsf{cnt}, H_j, \pi_{\mathsf{DDH}}^{3,j})$ or error from all P_j, $j \in Q \cap M$: Run all parties honestly

2.4. $(\mathsf{open}, eid||\mathsf{cnt})$ or error from all P_j, $j \in Q \cap M$: Call $A \subseteq Q \cap M$ the set of parties who sent error or got $\pi_{\mathsf{DDH}}^{3,j}$ rejected. If $A \neq \varnothing$, for all $i \in [N] \setminus M$ set

$$G \leftarrow \prod\nolimits_{j \in A} G_j, \quad r_A \leftarrow \sum\nolimits_{j \in A} r_j, \quad s \leftarrow \sum\nolimits_{j \in Q} s_j, \quad K_i \leftarrow k_i^{s r_A},$$

simulate $\pi_{\mathsf{DDH}}^{4,i}$ and send $(\mathsf{recon}, eid, K_i, \pi_{\mathsf{DDH}}^{4,i})$.

3. (reveal, eid) to P_i: Execute P_i normally
4. $(\mathsf{claim}, eid, \pi, m)$: Execute honest parties normally
5. a bit b: Return b.

Proof of Claim: First of all we remark that public and private parameters are correctly distributed because, calling $\sigma \in \mathbb{F}_q$ s.t. $h = g^\sigma$, the joint distribution $\sigma, (f_j)_{j \in M}$ is uniform over $\mathbb{F}_q^t$ and, since the interpolation is an isomorphism between $\mathbb{F}_q^t$ and $\mathbb{F}_q[x]_{<t}$, there exists a unique polynomial $f \sim U(\mathbb{F}_q[x]_{<t})$ such that $f(-1) = \sigma$ and $f(j) = f_j$. Moreover, by the properties of the Lagrange coefficients

$$k_i \;=\; h^{\lambda_{-1}} \prod\nolimits_{j \in M} k_j^{f_j} \;=\; \left[\lambda_{-1} f(-1) + \sum\nolimits_{j \in M} \lambda_j f(j)\right]_g \;=\; [f(i)]_g \,.$$

Next by the simulation soundness property, up to a negligible probability ε all the statements proved through an accepted NIZK argument are true. It follows that the reconstructions are correct because $G_j = \widetilde{G}^{r_j} = g^{s r_j} \;\Rightarrow$

$$\Rightarrow \quad G \;=\; \prod\nolimits_{j \in A} g^{s r_j} \;=\; [s r_A]_g \quad \Rightarrow \quad G^{f(i)} \;=\; g^{f(i) s r_A} \;=\; k_i^{s r_A} \;=\; K_i$$

Finally, under the above condition, $\mathcal{A}$ perfectly simulates H_0^ℓ when executed in DDH^0 while in DDH^1 it simulates $\mathsf{H}_0^{\ell-1}$ because calling $r_i = \rho^{-1}\beta \sim U(\mathbb{F}_q)$ then $\widehat{u}_1 = g^{r_i}$, $\widehat{u}_2 = h^{r_i}$. As a consequence in the ℓ-th election P_i's messages are of the form $R_i = g^{r_i}$, $\widetilde{c}_{1,i} = h^{r_i} \left[m_i \mathbf{a}^\top \mathbf{u}\right]_1$,

$$G_i = g^{(s+s_i) r_i} = \prod\nolimits_{j \in Q} \widetilde{G}_j^{r_i}, \quad H_i = h^{(s+s_i) r_i} = \prod\nolimits_{j \in Q} \widetilde{H}_j^{r_i}$$

It follows that $\mathcal{D}$ distinguishes the two world with advantage smaller than the advantage of $\mathcal{A}$ plus $\varepsilon(\lambda)$.

$\underline{\mathsf{H}_1^{\ell-1} \equiv \mathsf{H}_1^{\ell}}$: Given a PPTdistinguisher $\mathcal{D}$, we provide an algorithm $\mathcal{B}$ that breaks DDH_3 with almost the same advantage.
Description of $\mathcal{B}((\widehat{u}_1, \widehat{u}_2, g), \widehat{v}_0, (\widehat{v}_1, \widehat{v}_2, \widehat{v}_3))$: Run $M \xleftarrow{\$} \mathcal{D}$, wlog with $|M| = t-1$. It initializes a counter $\mathsf{ecnt} \leftarrow 0$, generates $\mathsf{mpk}, \mathsf{msk} \xleftarrow{\$} \mathsf{FE.Setup}(1^\lambda)$ and sk_i for all $i \in [N]$, samples $f \xleftarrow{\$} \mathbb{F}_q[x]_{<t}$ and evaluates $h \leftarrow g^{f(-1)}$, $k_i \xleftarrow{\$} g^{f(i)}$. Finally it calls $\mathsf{pp} \leftarrow (\mathsf{mpk}, g, h, k_0, \ldots, k_{N-1})$, $\mathsf{sp}_i \xleftarrow{\$} (\mathsf{sk}_i, f(i))$ and executes $\mathcal{D}$ until it returns:

1. (setup) as P_j: send $(\mathsf{input}, \mathsf{pp}, \mathsf{sp}_j)$ to P_j.

2. (elect, eid) or (toss, eid) from all parties: update $\mathsf{ecnt} \leftarrow \mathsf{ecnt}+1$ and simulate $\mathcal{F}_{\mathsf{CT}}^{\kappa,D}$ returning $(\mathsf{tossed}, eid, Q)$. If $Q \nsubseteq M$ let $i = \min(Q \setminus M)$, call

$$\begin{aligned}
&\text{If } \mathsf{ecnt} < \ell: && u_1, u_2, v_1, v_2, v_3 \xleftarrow{\$} \mathbb{G}_1 \\
&\text{If } \mathsf{ecnt} = \ell: && u_1, u_2, v_1, v_2, v_3 \leftarrow \widehat{u}_1, \widehat{u}_2, \widehat{v}_1, \widehat{v}_2, \widehat{v}_3 \\
&\text{If } \mathsf{ecnt} > \ell: && u_1, u_2 \xleftarrow{\$} \mathbb{G}_1,\ s_i \xleftarrow{\$} \mathbb{F}_q,\ v_0, v_1, v_2, v_3 \leftarrow g_1^{s_i}, u_1^{s_i}, u_2^{s_i}, g^{s_i}
\end{aligned}$$

   and set $v_4 \xleftarrow{\$} v_3^{f(-1)}$. Moreover in this case executes P_i setting $\widetilde{G}_i \leftarrow v_3$, $\widetilde{H}_i \leftarrow v_4$, $R_i \leftarrow u_1$, $\widetilde{c}_{1,i} \leftarrow u_2 \left[m_i \mathbf{a}^\top \mathbf{u}\right]_1$.

2.1. $(\mathsf{commit}, eid||0, \widetilde{G}_j, \widetilde{H}_j, R_j, \widetilde{c}_{1,j}, \pi_{\mathsf{Rng}}^j, \pi_{\mathsf{DDH}}^{0,j})$ or error from all P_j with $j \in Q \setminus M$: extract from the accepted arguments $\pi_{\mathsf{DDH}}^{0,j}$ and π_{Rng}^j values $s_j, r_j, m_j \in \mathbb{F}_q$ such that $\widetilde{G}_j = g^{s_j}$, $R_j = g^{r_j}$, $\widetilde{c}_{1,j} = h^{r_j}\left[m_j \mathbf{a}^\top \mathbf{u}\right]_1$.

2.2. $(\mathsf{open}, eid||0)$ or error from all P_j, $j \in M \cap Q$: keep executing P_i simulating its proofs, calling $s \leftarrow \sum_{j \in Q \setminus \{i\}} s_j$, $r \leftarrow \sum_{j \in Q \setminus \{i\}} r_j$, $m \leftarrow \sum_{j \in Q \setminus \{i\}} m_j$ and setting $G_i \leftarrow u_1^s v_1$, $H_i \leftarrow u_2^s v_2$,

$$\mathbf{c}_{0,i} \leftarrow [\mathbf{a}]_{v_0}, \quad c_{1,i} \leftarrow v_4^r v_2 \left[\mathbf{a}^\top((m+m_i)\mathbf{u} + \mathbf{w}_i)\right]_{v_0}, \quad c_{2,i} \leftarrow \left[\mathbf{a}^\top(\mathbf{w}_2 - \mathbf{u})\right]_{v_0}$$

2.3. $(\mathsf{msg}, eid||\mathsf{cnt}, \ldots)$ and $(\mathsf{commit}, eid||\mathsf{cnt}, \ldots)$ or error for all dishonest players in Q: Run all parties honestly.

2.4. $(\mathsf{open}, eid||\mathsf{cnt})$ or error from all dishonest players in Q: Run all parties honestly.

3. (reveal, eid) to P_i: Execute P_i normally.

4. $(\mathsf{claim}, eid, \pi, m)$: Execute honest parties normally.

5. a bit b: Return b.

Proof of Claim: this time $\mathcal{B}$ can run honest parties both on reconstruction phases and on reveal requests because initially it generates all the public and private parameters, not only the secret keys of the encryption scheme.

As before up to negligible probability ε, we assume that all the statements associated to accepted argument are true. Next, since $\mathsf{H}_1^{\ell-1}, \mathsf{H}_1^{\ell}$ differs only in the ℓ-th election, we show that when $\mathsf{ecnt} \neq \ell$, $\mathcal{B}$ sends to $\mathcal{D}$ the right distribution of messages. For $\mathsf{ecnt} < \ell$, P_i's message follows the right distribution by inspection. For $\mathsf{ecnt} > \ell$, P_i's initial messages are

$$\widetilde{G}_i = g^{s_i}, \quad \widetilde{H}_i = g^{f(-1)s_i} = h_i^s, \quad R_i = u_1, \quad \widetilde{c}_{1,i} = u_2 \left[m_i \mathbf{a}^\top \mathbf{u}\right]_1.$$

Analogously for the subsequent messages $G_i = u_1^s u_1^{s_i}$, $H_i = u_2^s u_2^{s_i}$, $\mathbf{c}_{0,i} = \left[s_i \mathbf{a}^\top \mathbf{u}\right]_1$, $c_{2,i} = \left[s_i \mathbf{a}^\top (\mathbf{w}_2 - \mathbf{u})\right]_1$ and

$$\begin{aligned} c_{1,i} &= h^{rs_i} u_2^{s_i} \left[\mathbf{a}^\top ((m + m_i)\mathbf{u} + \mathbf{w}_1)\right]_1 = \\ &= \left(\left[\mathbf{a}^\top \mathbf{w}_1\right]_1 \cdot u_2 \left[m_i \mathbf{a}^\top \mathbf{u}\right]_1 \cdot \prod\nolimits_{j \neq i} h^{r_j} \left[m_j \mathbf{a}^\top \mathbf{u}\right]_1\right)^{s_i} = \\ &= \left(\left[\mathbf{a}^\top \mathbf{w}_1\right]_1 \cdot \prod\nolimits_{j \in Q} \widetilde{c}_{1,j}\right)^{s_i} = \widetilde{c}_1^{s_i}. \end{aligned}$$

Therefore also in this case messages follows the right distribution.
Finally when $\mathcal{B}$ is executed in DDH_3^1, there exists an $s_i \sim U(\mathbb{F}_q)$ such that $(\widehat{v}_0, \widehat{v}_1, \widehat{v}_2, \widehat{v}_3) = (g_1, \widehat{u}_1, \widehat{u}_2, g)^{s_i}$ with $\widehat{u}_1, \widehat{u}_2 \sim U(\mathbb{G}_1)$. The same arguments used above shows that $\mathcal{B}$ perfectly simulates $\mathsf{H}_1^{\ell-1}$.
Conversely in DDH_3^0 all received values are random, therefore the ℓ-th election is carried out in the same way as the first $\ell - 1$ elections and consequently $\mathcal{B}$ simulates H_1^{ℓ}. It follows that $\mathcal{D}$'s advantage is smaller that $\mathcal{B}$'s plus ε.

$\underline{\mathsf{H}_2^{\ell-1} \equiv \mathsf{H}_2^{\ell}}$: To prove that the two game are equally distributed is enough to consider in the α-th election the bijection over $\mathbb{G}_1^2$:

$$(u_2, v_2) \mapsto \left(u_2 \left[-m_i \mathbf{a}^\top \mathbf{u}\right]_1, v_2 \left[-s_i \mathbf{a}^\top ((m + m_i)\mathbf{u} + \mathbf{w}_1)\right]_1\right).$$

Applying this map to u_2, v_2 in the ℓ-th election shows that the two game have the same distribution.

$\underline{\mathsf{H}_2^{L} \equiv \mathsf{H}_2^{\star}}$ We show that the two games are identically distributed through the perfect completeness of the NIZK argument used to prove $\mathcal{R}_{\mathsf{Dec}}$. In particular we need to prove that $(eid, j) \in E$ if and only, calling c the ciphertext produced during the election with ID eid,

$$\exists m \in \phi^{-1}(j) \ : \ \mathsf{FE.Dec}(c, (1, m), \mathsf{mpk}, \mathsf{sk}_{j,m}) \to 1.$$

If $(eid, j) \in E$ this means that c is the encryption of $(m, -1)$ with $j = \phi(m)$. In particular the key $\mathsf{sk}_{j,m}$ decrypts c because $(m, -1)^\top (1, m) = 0$. Moreover this key is generated through the key derivation algorithm, therefore $(\mathsf{mpk}, (1, m), \mathsf{sk}_{j,m}) \in \mathcal{L}_{\mathsf{key}}$. In conclusions $\pi \xleftarrow{\$} \mathsf{NIZK.P}_{\mathsf{Dec}}(\mathsf{mpk}, c, (1, m), \mathsf{sk}_{j,m})$ is later accepted since the arguments belong to $\mathcal{R}_{\mathsf{Dec}}$.
Conversely, assuming that there exists a key $\mathsf{sk}_{j,m}$ that decrypts c, calling $(m', -1)$ by correctness $(m', -1)^\top (1, m) = 0$ that is $m = m'$ and therefore $(eid, j) = (eid, \phi(m')) \in E$.

$\underline{\mathsf{H}_3^{\ell-1} \equiv \mathsf{H}_3^{\ell}}$: Given a PPTdistinguisher $\mathcal{D}$ we build an algorithm $\mathcal{C}$ that breaks the selective security of the underlying FE scheme.

Description of $\mathcal{C}$: Initially runs $M \leftarrow \mathcal{D}$, samples $\bar{m}_0 \xleftarrow{\$} \phi^{-1}([N] \setminus M)$ and $\bar{m}_1 \xleftarrow{\$} \mathbb{F}_q \setminus \phi^{-1}(M)$. Sends $(\bar{m}_0, -1), (\bar{m}_1, -1)$ to the challenger and waits for his reply $(\mathsf{mpk}, \widehat{\mathbf{c}}_0, \widehat{c}_1, \widehat{c}_2)$, then requests secret keys $\mathsf{sk}_{\phi(m),m}$ for vectors $(1, m)$ with $m \in \phi^{-1}(M)$. Initializes a counter $\mathsf{ecnt} \leftarrow 0$, and two sets $B, E \leftarrow \varnothing$, samples $f \xleftarrow{\$} \mathbb{F}_q[x]_{<t}$, $g \xleftarrow{\$} \mathbb{G}_1$ and computes $h = g^{f(-1)}$, $k_i = g^{f(i)}$. Called $\mathsf{pp} \leftarrow (\mathsf{mpk}, g, h, k_0, \ldots, k_{N-1})$ and $\mathsf{sp}_j \leftarrow (\mathsf{sk}_j, f(j))$, runs $\mathcal{D}$ until it returns:

1. (setup) as P_j: send $(\mathsf{input}, \mathsf{pp}, \mathsf{sp}_j)$ to P_j.

2. (elect, eid) from all honest parties: update $\mathsf{ecnt} \leftarrow \mathsf{ecnt}{+}1$ and simulate $\mathcal{F}_{\mathsf{CT}}^{\kappa,D}$ returning $(\mathsf{tossed}, eid, Q)$. If $Q \nsubseteq M$ call $i = \min(Q \setminus M)$, sample $u_1, u_2, v_1, v_2, v_3 \xleftarrow{\$} \mathbb{G}_1$ and set $v_4 = v_3^{f(-1)}$. Execute P_i simulating its proof and setting

$$\widetilde{G}_i \leftarrow v_3, \quad \widetilde{H}_i \leftarrow v_4, \quad R_i \leftarrow u_1, \quad \widetilde{c}_{1,i} \leftarrow u_2$$

2.1. $(\mathsf{commit}, eid||0, \widetilde{G}_j, \widetilde{H}_j, R_j, \widetilde{c}_{1,j}, \pi^j_{\mathsf{Rng}}, \pi^{0,j}_{\mathsf{DDH}})$ or error from all P_j with $j \in Q \setminus M$: extract from the accepted arguments s_j, r_j, m_j.

2.2. $(\mathsf{open}, eid||0)$ or error from all P_j, $j \in M \cap Q$: Keep executing P_i simulating its arguments, calling $s \leftarrow \sum_{j \in Q\setminus\{i\}} s_j$, $r \leftarrow \sum_{j\in Q\setminus\{i\}} r_j$ and setting $G_i \leftarrow u_1^s v_1$, $c_{1,i} \leftarrow v_4^r v_2$,

$$\begin{aligned} &\text{If } \mathsf{ecnt} \neq \ell \quad \mathbf{c}_{0,i} \leftarrow [s_i \mathbf{a}]_1\,,\; c_{2,i} \leftarrow \left[s_i \mathbf{a}^\top (\mathbf{w}_2 - \mathbf{u})\right]_1 \\ &\text{If } \mathsf{ecnt} = \ell \quad \mathbf{c}_{0,i} \leftarrow \widehat{\mathbf{c}}_0,\; c_{2,i} \leftarrow \widehat{c}_{2,i} \end{aligned}$$

2.3. $(\mathsf{msg}, eid||\mathsf{cnt}, G_j, \mathbf{c}_{0,j}, c_{1,j}, c_{2,j}, \pi^{1,j}_{\mathsf{DDH}}, \pi^{2,j}_{\mathsf{DDH}})$ and $(\mathsf{commit}, eid||\mathsf{cnt}, H_j, \pi^{3,j}_{\mathsf{DDH}})$ or error from all P_j, $j \in Q \cap M$: Run all parties honestly.

2.4. $(\mathsf{open}, eid||\mathsf{cnt})$ or error from all P_j, $j \in Q \cap M$. Add $B \leftarrow B \cup \{(eid, m)\}$.
If $\mathsf{ecnt} < \ell$: If $\bar{m}_0 - m \notin [N]$ return a random bit and halt. Otherwise sample $j^* \xleftarrow{\$} [N]$ and add $E \leftarrow (eid, j^*)$. Find the only $m_i \in [N]$ such that $\phi(m + m_i) = j^*$ and set

$$\begin{aligned} &\text{If } j^* \notin M: \quad H_i \xleftarrow{\$} \mathbb{G}_1. \\ &\text{If } j^* \in M: \quad H_i \leftarrow u_2^s v_2 \left[-s_i \mathbf{a}^\top ((m + m_i)\mathbf{u} + \mathbf{w}_1)\right]_1 \left[-s m_i \mathbf{a}^\top \mathbf{u}\right]_1. \end{aligned}$$

If $\mathsf{ecnt} = \ell$: sample $j^* \xleftarrow{\$} [N]$, find the only $m_i \in [N]$ such that $\phi(m + m_i) = j^*$ and set

$$\begin{aligned} &\text{If } j^* \notin M: \quad j \leftarrow \phi(\bar{m}_0),\; H_i \leftarrow u_2^s v_2 \cdot \widehat{c}_1^{-1} \left[-s(\bar{m}_0 - m)\mathbf{a}^\top \mathbf{u}\right]_1 \\ &\text{If } j^* \in M: \quad j \leftarrow j^*,\; H_i \leftarrow u_2^s v_2 \left[-s_i \mathbf{a}^\top ((m + m_i)\mathbf{u} + \mathbf{w}_1)\right]_1 \left[-s m_i \mathbf{a}^\top \mathbf{u}\right]_1. \end{aligned}$$

Finally add $E \leftarrow E \cup \{(eid, j)\}$.
If $\mathsf{ecnt} > \ell$: add $E \leftarrow E \cup \{(eid, \phi(m + m_i))\}$ and set H_i normally, i.e.

$$H_i \leftarrow u_2^s v_2 \left[-s_i \mathbf{a}^\top ((m + m_i)\mathbf{u} + \mathbf{w}_1)\right]_1 \left[-s m_i \mathbf{a}^\top \mathbf{u}\right]_1.$$

Send $i' \in [N] \setminus \phi^{-1}(M)$ send to $P_{i'}$ (lost, eid) if $i' \neq j$ or (won, eid) otherwise.

3. (reveal, eid) to P_i: If $(eid, i) \in E$ and $(eid, m) \in B$ find $m' \in [N]$ such that $\phi(m+m') = i$, simulate π and send $(\mathsf{claim}, eid, \pi, m+m')$ to dishonest players and $(\mathsf{results}, eid, i)$ from honest ones.

4. $(\mathsf{claim}, eid, \pi, m)$ as P_j: Execute honest parties normally.

5. a bit b: Return b.

Proof of Claim: Initially observe that the coinciding phases of $\mathsf{H}_3^{\ell-1}$ and H_3^{ℓ} are performed correctly by $\mathcal{C}$. First of all the public parameters are correct by inspection, while all the requests to $\mathsf{sk}_{\phi(m),m}$ for $m \in \phi^{-1}(M)$ are accepted by the challenger because both $\bar{m}_1, \bar{m}_2 \notin \phi^{-1}(M)$, therefore

$$(\bar{m}_0, -1)^\top (1, m) = \bar{m}_0 - m \neq 0 \qquad (\bar{m}_1, -1)^\top (1, m) = \bar{m}_1 - m \neq 0.$$

Also reveal and claim requests are handled correctly as in H_2^*.
Moving to the election phases, all the steps but the last one follows the right distribution. This is clearly true when $\mathsf{ecnt} \neq \ell$. To deal with the case $\mathsf{ecnt} = \ell$, let us call β the plaintext chosen by the challenger and $s_i \sim U(\mathbb{F}_q)$ such that

$$\widehat{\mathbf{c}}_0 = [s_i \mathbf{a}]_1, \quad \widehat{c}_1 = \left[s_i \mathbf{a}^\top (\bar{m}_\beta \mathbf{u} + \mathbf{w}_1)\right]_1, \quad \widehat{c}_2 = \left[s_i \mathbf{a}^\top (\mathbf{w}_2 - \mathbf{u})\right]_1.$$

Then it follows that even when $\mathsf{ecnt} = \ell$, $\mathbf{c}_{0,i}$ and $c_{2,i}$ are correctly distributed. Regarding the last step instead when $\mathsf{ecnt} \neq \ell$, the generation of H_i and the update of B, E is performed as prescribed in $\mathsf{H}_3^{\ell-1}$, H_3^{ℓ}. Next we show that $\Pr[\bar{m}_0 - m \in [N]] = 1/\kappa$.

$$\Pr[\bar{m}_0 \in m + [N]] \;=\; \frac{|(m+[N]) \setminus \phi^{-1}(M)|}{|[N] \setminus \phi^{-1}(M)|} \;=\; \frac{N-T}{\kappa(N-T)} \;=\; \frac{1}{\kappa^{-1}}.$$

When $\beta = 0$ and the check $\bar{m}_0 - m \in [N]$ passes, $\mathcal{C}$ perfectly simulates $\mathsf{H}_3^{\ell-1}$ because in the ℓ-th election, it produces H_i of the form

$$u_2^s v_2 \left[-s_i \mathbf{a}^\top ((m+m_i)\mathbf{u} + \mathbf{w}_1)\right]_1 \left[-s_i (m_i) \mathbf{a}^\top \mathbf{u}\right]_1$$

with $m_i = \bar{m}_0 - m \in [N]$ when $j^* \notin M$ and $m_i \in [N]$ such that $\phi(m_i + m) = j^*$ when $j^* \in M$. To conclude we need to show that $m_i \sim U([N])$ or equivalently that $m_i + m \sim U([N] + m)$.
To prove the latter, for all $a \in [N] + m$, if $\phi(a) \notin M$ then

$$\begin{aligned}
\Pr[m_i + m = a] \;&=\; \Pr[\bar{m}_0 = a | \bar{m}_0 - m \in [N],\ j^* \notin M] \Pr[j^* \notin M] \;+ \\
&\quad +\; \Pr[m + m_i = a | j^* \in M] \Pr[j^* \in M] \;= \\
&=\; \frac{\Pr[\bar{m}_0 = a] \Pr[j^* \notin M]}{\Pr[\bar{m}_0 - m \in [N]]} \;=\; \frac{\kappa}{\kappa(N - |M|)} \cdot \frac{N - |M|}{N} \;=\; \frac{1}{N}
\end{aligned}$$

where in the second equality, the second term is zero because if $m + m_j = a$ then $j^* = \phi(a) \notin M$. Conversely if $\phi(a) \in M$ then

$$\begin{aligned}
\Pr[m + m_i = a] \;&=\; \Pr[\bar{m}_0 = a | \bar{m}_0 - m \in [N],\ j^* \notin M] \Pr[j^* \notin M] \;+ \\
&\quad +\; \Pr[m + m_i = a | j^* \in M] \Pr[j^* \in M] \;= \\
&=\; \Pr[\phi(a) = j^* | j^* \in M] \Pr[j^* \in M] \;=\; \Pr[\phi(a) = j^*] \;=\; \frac{1}{N}
\end{aligned}$$

where the first term is zero because $\phi(a) \in M$ but $\phi(\bar{m}_0) \notin M$. Moreover the first equality holds because $j^* \in M$ implies that $m + m_i = a \in [N] + m$ if and only if $\phi(a) = j^*$. Therefore under the above conditions $\mathcal{C}$ simulates $\mathsf{H}_3^{\ell-1}$. As a consequence, called ck the event $\bar{m}_0 - m \in [N]$ and $E^(\ell - 1)$ the event that $\mathcal{D}$ returns 0 when executed with $\mathsf{H}_3^{\ell-1}$ then

$$\begin{aligned}\Pr[\mathcal{C} \to 0|\beta = 0] &= \Pr[\mathcal{C} \to 0|\beta = 0,\ \mathsf{ck}]\Pr[\mathsf{ck}] \ + \ \Pr[\mathcal{C} \to 0|\beta = 0,\ \neg\mathsf{ck}]\Pr[\neg\mathsf{ck}] \\ &= \Pr[E(\ell-1)]\frac{1}{\kappa} + \frac{\kappa - 1}{2\kappa}.\end{aligned}$$

Conversely when $\beta = 1$, we claim that $\mathcal{C}$ produces a distribution statistically close to the one $\mathcal{D}$ observes in H_3^ℓ. With our previous considerations in mind it is enough to prove, recalling $\mathbf{a}^\top\mathbf{u} \neq 0$ and $s_i \neq 0$, that upon conditioning on all other messages sent by $\mathcal{C}$, H_i in the ℓ-th election is statistically close to a uniformly distributed H_i^*. We remind that in this case H_i^* is equal to

$$\begin{aligned}H_i &= u_2^s v_2 \left[-s_i\mathbf{a}^\top(\bar{m}_1\mathbf{u} + \mathbf{w}_1)\right]_1 \left[-s(\bar{m}_0 - m)\mathbf{a}^\top\mathbf{u}\right]_1 \\ &= u_2^s v_2 \left[-s_i\mathbf{a}^\top\mathbf{w}_1 - s(\bar{m}_0 - m)\mathbf{a}^\top\mathbf{u}\right]_1 \left[-s_i\mathbf{a}^\top\mathbf{u}\right]_1^{\bar{m}_1}.\end{aligned}$$

By Proposition 1 to conclude we only have to show that $\left[-s_i\mathbf{a}^\top\mathbf{a}\right]_1^{\bar{m}_1}$ is statistically close to the uniform distribution. This follows as exponentiating a fixed generator, in this case $\left[-s_i\mathbf{a}^\top\mathbf{u}\right]_1$, defines a bijection between $\mathbb{F}_q$ and $\mathbb{G}_1$. Now, $\bar{m}_1 \sim U(\mathbb{F}_q \setminus \phi^{-1}(M))$ so given a uniformly random $x \sim \mathbb{F}_q$

$$\varDelta(\bar{m}_1, x) \ = \ \frac{|\phi^{-1}(M)|}{q} \ = \ \frac{\kappa T}{q}$$

that is negligible. It follows that the claim is true and in particular the statistical distance between the view simulated by $\mathcal{C}$ and the actual view of H_3^ℓ by Proposition 5 is smaller than $\varepsilon = (\kappa T)q^{-1}$. Calling $E(\ell)$ the event $\mathcal{D}$ returns 0 in H_3^ℓ and $\widetilde{E}(\ell)$ the event $\mathcal{D}$ returns 0 when executed by $\mathcal{D}$ with $\beta = 1$ and $\bar{m}_0 - m \in [N]$ then by Propositions 2 and 4

$$\left|\Pr[E(\ell)] - \Pr\left[\widetilde{E}(\ell)\right]\right| \leq \varepsilon.$$

Now, with the same steps used in the case $\beta = 0$

$$\begin{aligned}\Pr[\mathcal{C} \to 0|\beta = 1] &= \Pr[\mathcal{C} \to 0|\beta = 1,\ \mathsf{ck}]\Pr[\mathsf{ck}] + \Pr[\mathcal{C} \to 0|\beta = 1,\ \neg\mathsf{ck}]\Pr[\neg\mathsf{ck}] \\ &= \Pr\left[\widetilde{E}(\ell)\right]\frac{1}{\kappa} + \frac{\kappa - 1}{2\kappa}\end{aligned}$$

In conclusion

$$\begin{aligned}|\Pr[E(\ell-1)] - \Pr[E(\ell)]| &\leq \left|\Pr[E(\ell-1)] - \Pr\left[\widetilde{E}(\ell)\right]\right| + \left|\Pr\left[\widetilde{E}(\ell)\right] - \Pr[E(\ell)]\right| \\ &\leq \kappa\left|\Pr[\mathcal{C} \to 0|\beta = 0] - \Pr[\mathcal{C} \to 0|\beta = 1]\right| + \varepsilon\end{aligned}$$

Therefore the advantage of $\mathcal{D}$ is negligible.

$\underline{\mathsf{H}_3^L \equiv \mathsf{H}_4}$: Every time a malicious P_j request $(\mathsf{claim}, eid, \pi, m)$ such that $\phi(m) = j$ and π is accepted by $\mathsf{NIZK.V_{Dec}}$ then $(eid, j) \in E$. Let c be the ciphertext produced in the election eid. From the simulation soundness of π, up to negligible probability $\mathsf{NIZK.V_{Dec}}(\mathsf{mpk}, c, (1, m), \pi) \to 1$ implies that there exists sk with $(\mathsf{mpk}, \mathsf{sk}, (1, m)) \in \mathcal{L}_{\mathsf{key}}$ such that

$$\mathsf{FE.Dec}(c, (1, m), \mathsf{mpk}, \mathsf{sk}) = 1.$$

From the definition of $\mathcal{L}_{\mathsf{key}}$, calling $(m', -1)$ the message encrypted in c, $(m', -1)^\top(1, m) = 0$ that is $m = m'$ and in particular $\phi(m') = \phi(m) = j$. If $(eid, j^*) \in E$ with $j^* \notin M$ and if the committee wasn't fully dishonest then m' was chosen randomly over $\mathbb{F}_q$, even though this happens with probability

$$\Pr\left[\phi(m') = j\right] \;=\; \frac{|\phi^{-1}(j)|}{q} \;=\; \frac{\kappa}{q}$$

that is negligible. Conversely if a dishonest committee produced c, then $\phi(m') = j^*$ implies $j^* = j$. Finally if $j^* \in M$ then by construction $\phi(m') = j^*$ which again implies $j^* = j$ and in particular $(eid, j) \in E$ as claimed.

$\underline{\mathsf{H}_4 \equiv \mathcal{S} \circ \mathcal{F}^{\kappa}_{\mathsf{SSLE}}}$: Follows by inspection.

D.4 Proof of Theorem 3

Proof. As in the proof of Theorem 2 we need to show that there exists a simulator such that $\mathcal{S} \circ \mathcal{F}_{\mathsf{Setup}}$ is indistinguishable from the real protocol for any $\mathcal{Z}$ satisfying the hypothesis.

Description of $\mathcal{S}$: Initially wait for $M \xleftarrow{\$} \mathcal{Z}$. For all $i \in [N] \setminus M$ sample $\mathbf{x}_i \leftarrow \mathbb{F}_q^2$, set $\widehat{\mathbf{h}}_i \leftarrow [\mathbf{x}_i]_2$, $\pi^i_{\mathsf{Lin}} \leftarrow \mathsf{NIZK.P_{Lin}}(g_2, \widehat{\mathbf{h}}_i, \mathbf{x}_i)$ and broadcast $(\mathsf{recepit}, \perp, i)$. Wait for $(\mathsf{commit}, \perp, \widehat{\mathbf{h}}_j, \pi^j_{\mathsf{Lin}})$ or error from $\mathcal{Z}$ on behalf of P_j for all $j \in M$ and broadcast $(\mathsf{decom}, \perp, i, \widehat{\mathbf{h}}_j, \pi^j_{\mathsf{Lin}})$. Finally wait for $(\mathsf{open}, \perp)$ from $\mathcal{Z}$ on behalf of P_j for all $j \in M$ and add to D player who sent error or got π^j_{Lin} rejected. Upon receiving:

1. $(\mathsf{share\_request}, sid)$ from P_j: Send (setup, sid) to $\mathcal{F}_{\mathsf{Setup}}$. When it replies $(\mathsf{input}, \mathsf{pp}, \mathsf{sp}_j)$ with $\mathsf{pp} = \mathsf{msk}, g, h, (s_i)_{i=0}^{N-1}, f(j)$, send $(\mathsf{share}, sid, g, h, (s_i)_{i=0}^{N-1}, f(j))$ to P_j.

2. A request to send (input, sid) to P_i form $\mathcal{F}_{\mathsf{Setup}}$: Stop the message. When this has happened for all honest parties, send (setup, sid) to $\mathcal{F}_{\mathsf{Setup}}$ for all $j \in M$ and wait for $(\mathsf{input}, \mathsf{pp}, \mathsf{sp}_j)$. Parse $\mathsf{pp} = \mathsf{mpk}, g, h, (s_i)_{i=0}^{N}$ and $\mathsf{sp}_i = \mathsf{sk}_j, f(j)$ with

$$\mathsf{mpk} = \widehat{\mathbf{k}}, \widehat{k}_0, \widehat{k}_1, \widehat{k}_2, \quad \mathsf{sk}_j = (\mathsf{sk}_{j,m})_{m \in \phi^{-1}(j)}, \quad \mathsf{sk}_{j,m} = (\widehat{\mathbf{d}}_m, \widehat{d}_m)$$

   Sample $R \xleftarrow{\$} \{0,1\}^\lambda$, $\widehat{d}_m \xleftarrow{\$} \mathbb{G}_2$ for all $m \in [\kappa N] \setminus \phi^{-1}(M)$ and program $\mathcal{H}(R) = (\widehat{\mathbf{k}}, \widehat{k}_0, d_m)_{m \in [\kappa N]}$.
   Simulate $\mathcal{F}^{\lambda, D, 1}_{\mathsf{CT}}$ returning $(\mathsf{tossed}, sid, Q, R)$ and abort if $Q \subseteq M$.
   Set $i = \min(Q \setminus M)$, simulate all parties honestly besides P_i and broadcast $(\mathsf{recepit}, sid, i)$ from both $\mathcal{F}_{\mathsf{Com}}$ and $\mathcal{F}^{\mathsf{Lin}}_{\mathsf{Czk}}$

2.1. $(\mathsf{commit}, sid, G_{j,\psi(m)}, C_{j,m}, \pi^{j,\psi(m)}_{\mathsf{DDH}})$, $(\mathsf{prove}, sid, \mathbf{k}, (k_{j,1}, k_{j,2}), (\mathbf{w}_{j,1}, \mathbf{w}_{j,2}))$ or error from P_j, $\forall j \in Q \cap M$: Run P_i sampling $\mathbf{w}_{i,1}, \mathbf{w}_{i,2} \xleftarrow{\$} \mathbb{F}_q^2$, $r_{i,\psi(m)} \xleftarrow{\$} \mathbb{F}_q$ and setting

$$k_{i,1} \leftarrow \widehat{k}_1 \cdot \mathbf{k}^{\mathbf{w}_{i,1}}, \quad k_{i,2} \leftarrow \widehat{k}_2 \cdot \mathbf{k}^{\mathbf{w}_{i,2}},$$

$$\mathbf{d}_{i,m} \leftarrow \widehat{\mathbf{d}}_m \cdot [\mathbf{w}_{i,1} + m\mathbf{w}_{i,2}]_{d_m}, \quad G_{i,\psi(m)} \leftarrow [r_{i,\psi(m)}]_2$$

Moreover for $m \in \phi^{-1}(M \setminus D)$ compute $C_{i,m} \leftarrow \widehat{\mathbf{h}}_{\phi}(m)^{r_{i,\psi(m)}}\mathbf{d}_{i,m}$ and for $m \notin \phi^{-1}(M)$, $C_{i,m} \leftarrow^{\$} \mathbb{G}_2^2$.

2.2. (open, sid) to $\mathcal{F}_{\mathsf{Com}}, \mathcal{F}_{\mathsf{Czk}}^{\mathsf{Lin}}$ or error from P_j for all $j \in Q \cap M$: Remove from Q player that sent error and simulate $\mathcal{F}_{\mathsf{Com}}$, $\mathcal{F}_{\mathsf{Czk}}^{\mathsf{Lin}}$. Compute $\bar{\mathbf{w}}_1 = -\sum_{j\in Q} \mathbf{w}_{j,1}$ and $\bar{\mathbf{w}}_2 = -\sum_{j \in Q} \mathbf{w}_{j,2}$ and simulate $\mathcal{F}_{\mathsf{CT}}^{4,\mathbb{F}_q}$ returning $(\mathsf{tossed}, sid, \bar{\mathbf{w}}_1, \bar{\mathbf{w}}_2)$.
Execute all player honestly from now on using $(\mathbf{x}_i)_{i\in[N]\setminus M}$ to decrypt.

2.3. $(\mathsf{accept}, sid, j')$ or error from P_j, for all $j \in M$ and $j' \in Q$: Allow $\mathcal{F}_{\mathsf{Setup}}$ to send (input, sid).

To prove the Theorem we go through a sequence of hybrid games. Let L be an upper bound on the number of setups requested by the environment.

- $\mathsf{H}_{\mathsf{real}}$: the real protocol
- H_0: as $\mathsf{H}_{\mathsf{real}}$ but if any call to $\mathcal{F}_{\mathsf{CT}}^{\lambda,D,1}$ return $(\mathsf{tossed}, sid, Q, R)$ with $R \subseteq M$, ignore any further message
- $\mathsf{H}_0^0 = \mathsf{H}_0$, $\mathsf{H}_m^{L+1} = \mathsf{H}_{m+1}^0$. $\mathsf{H}_m^{\ell+1}$: as H_m^{ℓ} but in the $\ell+1$-th instance of $\mathcal{F}_{\mathsf{Setup}}$, calling Q, R the message sent by $\mathcal{F}_{\mathsf{CT}}^{\lambda,D,1}$ and $i = \min(Q \setminus M)$ if $\phi(m) \notin M$ set $C_{i,m} \leftarrow^{\$} \mathbb{G}_2^2$
- H_1: as $\mathsf{H}_{\kappa N}^{L+1}$ but k_0, returned by the random oracle, is uniformly distributed over $\mathbb{G}_1 \setminus \{1_{\mathbb{G}_1}\}$.
- H_2: as H_1 in all executions of $\mathcal{F}_{\mathsf{Setup}}$, calling Q, R the message sent by $\mathcal{F}_{\mathsf{CT}}^{\lambda,D,1}$ and $i = \min(Q \setminus M)$, P_i sample $\bar{\mathbf{w}}_1, \bar{\mathbf{w}}_2 \leftarrow^{\$} \mathbb{F}_q^2$ and set

$$k_{i,1} \leftarrow \mathbf{k}^{\bar{\mathbf{w}}_1 + \mathbf{w}_{i,1}}, \quad k_{i,2} \leftarrow \mathbf{k}^{\bar{\mathbf{w}}_2 + \mathbf{w}_{i,2}}, \quad \mathbf{d}_{i,m} \leftarrow [\bar{\mathbf{w}}_1 + \mathbf{w}_1 + m(\bar{\mathbf{w}}_2 + \mathbf{w}_2)]_{d_m}$$

Moreover calling $\mathbf{w}_{j,1}, \mathbf{w}_{j,2}$ the vectors received by $\mathcal{F}_{\mathsf{Czk}}^{\mathsf{Lin}}$, then $\mathcal{F}_{\mathsf{CT}}^{4,\mathbb{F}_q}$ returns

$$-\sum_{j\in Q} \mathbf{w}_{j,1}, \quad -\sum_{j \in Q} \mathbf{w}_{j,2}.$$

$\mathsf{H}_{\mathsf{real}} \equiv \mathsf{H}_0$: By construction the event $Q \subseteq M$ happens with probability $2^{-\lambda}$ at most once per instance of $\mathcal{F}_{\mathsf{Setup}}$. Using a union bound the probability that H_0 halts is smaller that $L/2^{\lambda}$, where L is polynomially bounded.

$\mathsf{H}_{m^*}^{\ell-1} \equiv \mathsf{H}_{m^*}^{\ell}$: Given a distinguisher $\mathcal{D}$ we can define $\mathcal{A}$ that breaks the DDH_3 with the same advantage.
Description of $\mathcal{A}$: Initially wait for the challenger to send $(g_2, \mathbf{u}), (v, \mathbf{v}) \in \mathbb{G}_2^3$ and $M \leftarrow^{\$} \mathcal{D}$. Call $i^* = \phi(m^*)$ and if $i^* \in M$ then simulate $\mathsf{H}_{m^*}^{\ell-1}$ until $\mathcal{D}$ halts. Otherwise for all $i \in [N] \setminus M$, $i \neq i^*$ sample $\mathbf{x}_i \leftarrow^{\$} \mathbb{F}_q^2$ compute $\widehat{\mathbf{h}}_i \leftarrow [\mathbf{x}_i]_2$ while for i^* set $\widehat{\mathbf{h}}_{i^*} \leftarrow \mathbf{u}$. For all $i \notin M$ simulate π_{Lin}^i. Broadcast $(\mathsf{recepit}, \perp, i)$ and wait for $(\mathsf{commit}, \perp, \widehat{\mathbf{h}}_j, \pi_{\mathsf{Lin}}^j)$ or error from $\mathcal{D}$ on behalf of P_j.
Extract $\mathbf{x}_j$ from accepting proofs and broadcast $(\mathsf{decom}, \perp, i, \widehat{\mathbf{h}}_i, \pi_{\mathsf{Lin}}^i)$ for $i \in M$. Finally wait for $(\mathsf{open}, \perp)$ or error from $\mathcal{D}$ on behalf of P_j, $j \in M$ and add to D those who sent error or a rejected proof.
Next initialise $\mathsf{scnt} \leftarrow 0$ and sample $g \leftarrow^{\$} \mathbb{G}_1$, $f \leftarrow^{\$} \mathbb{F}_q[x]_{<t}$ and compute $h \leftarrow g^{f(-1)}$, $s_i \leftarrow g^{f(i)}$. Run $\mathcal{D}$ until it returns:

1. $(\mathsf{share\_request}, sid)$ from corrupted user P_j: Simulate $\mathcal{F}_{\mathsf{VSS}}$ returning the tuple $\Big(\mathsf{share}, sid, g, h, (s_i)_{i=0}^{N-1}, f(j)\Big)$ to P_j.

2. (setup, sid) from all honest users: Add $\mathsf{scnt} \leftarrow \mathsf{scnt}{+}1$ and simulate $\mathcal{F}_{\mathsf{CT}}^{\lambda,D,1}$ returning $(\mathsf{tossed}, sid, Q, R)$. If $Q \subseteq M$ halt and return a random bit, otherwise call $i = \min(Q \setminus M)$, simulate all users honestly and broadcast $(\mathsf{recepit}, sid, i')$ from $\mathcal{F}_{\mathsf{Com}}, \mathcal{F}_{\mathsf{Czk}}^{\mathsf{Lin}}$ for all $i' \in Q \setminus M$.

2.1. $(\mathsf{commit}, sid, G_{j,\psi(m)}, C_{j,m}, \pi_{\mathsf{DDH}}^{j,\psi(m)}), (\mathsf{prove}, sid, \mathbf{k}, (k_{j,1}, k_{j,2}), (\mathbf{w}_{j,1}, \mathbf{w}_{j,2}))$ or error from P_j for all $j \in Q \cap M$: Extract from accepting arguments $r_{j,\psi(m)}$ and simulate all users and functionalities correctly besides P_i. Run P_i setting

$$\begin{array}{ll} \text{If } \mathsf{scnt} \neq \ell \ \vee \ \psi(m) \neq \psi(m^*) & G_{i,\psi(m)} \leftarrow \left[r_{i,\psi(m)}\right]_2 \\ \text{If } \mathsf{scnt} = \ell \ \wedge \ \psi(m) = \psi(m^*) & G_{i,\psi(m)} \leftarrow v \end{array}$$

Moreover for $m < m^*$ and $\phi(m) \notin M$ set $C_{i,m} \xleftarrow{\$} \mathbb{G}_2^2$; for $m > m^*$ with $\psi(m) = \psi(m^*)$ and $\mathsf{scnt} = \ell$ set $C_{i,m} \leftarrow v^{\mathbf{x}_{\phi(m)}} \mathbf{d}_{i,m}$; for m^*

$$\begin{array}{ll} \text{If } \mathsf{scnt} < \ell & C_{i,m^*} \xleftarrow{\$} \mathbb{G}_2^2 \\ \text{If } \mathsf{scnt} = \ell & C_{i,m^*} \leftarrow \mathbf{v} \cdot \mathbf{d}_{i,m^*} \\ \text{If } \mathsf{scnt} > \ell & C_{i,m^*} \leftarrow \mathbf{u}^{r_{i,\psi(m^*)}} \mathbf{d}_{i,m^*}. \end{array}$$

In all other cases $C_{i,m} = \widehat{\mathbf{h}}_{\phi(m)}^{r_{i,\psi(m)}} \mathbf{d}_{i,m}$.

2.2. (open, sid) for $\mathcal{F}_{\mathsf{Com}}, \mathcal{F}_{\mathsf{CT}}^{\lambda,D,1}$ or error from P_j for all $j \in Q \cap M$: Execute all uncorrupted users honestly but P_{i^*}. Run P_{i^*} setting $\mathbf{d}_{j,m} \leftarrow \mathbf{u}^{-r_{j,\psi(m)}} \cdot C_{j,m}$ for $m \in \phi^{-1}(i^*)$

3. a bit b: Return b.

Proof of Claim: We by showing that regardless of the experiment $\mathcal{A}$ is executed in, he simulates well the execution of P_{i^*} after the commitment's opening. To see this assume up to negligible probability that all the statement proved through accepted NIZK are correct, then the ciphertexts produced by P_j are of the form, for $m \in \phi^{-1}(i^*)$

$$G_{j,m} = \left[r_{i,\psi(m)}\right]_2, \quad C_{j,m} \ = \ \widehat{\mathbf{h}}_{i^*}^{r_{i,\psi(m)}} \cdot \mathbf{d}_{i,m} \ = \ \mathbf{u}^{r_{i,\psi(m)}} \cdot \mathbf{d}_{i,m}$$

therefore P_{i^*} extract the right value. Next, if $\mathcal{A}$ is executed in DDH_3^1 then there exists $r_{i,\psi(m^*)} \sim U(\mathbb{F}_q)$ such that $v = u^{r_{i,\psi(m^*)}}$, $\mathbf{v} = \mathbf{u}^{r_{i,\psi(m^*)}}$. Therefore we observe that ciphertext $C_{i,m}$ with $m < m^*$ associated to uncorrupted users are well formed, since they are random. For $m > m^*$ the only case we need to check is when $\psi(m) = \psi(m^*)$ and $\mathsf{snct} = \ell$. Under this conditions

$$C_{i,m} \ = \ v^{\mathbf{x}_{\phi(m)}} \mathbf{d}_{i,m} \ = \ \Big[r_{i,\psi(m) \cdot \mathbf{x}_{\phi(m)}}\Big]_2 \cdot \mathbf{d}_{i,m} \ = \ \widehat{\mathbf{h}}_{\phi}(m)^{r_{i,\psi(m)}} \mathbf{d}_{i,m}.$$

Finally for m^*, when $\mathsf{snct} < \ell$ the ciphertext is random as expected and the same is true for $\mathsf{scnt} > \ell$ because $\mathbf{u} = \widehat{\mathbf{h}}_{i^*}$ by construction. Finally when $\mathsf{scnt} = \ell$

$$C_{i,m^*} \ = \ \mathbf{v} \cdot \mathbf{d}_{i,m^*} \ = \ \mathbf{u}^{r_{i,\psi(m^*)}} \cdot \mathbf{d}_{i,m^*} \ = \ \widehat{\mathbf{h}}_{i^*}^{r_{i,\psi(m^*)}} \cdot \mathbf{d}_{i,m^*}.$$

In conclusion $\mathcal{A}$ simulates $\mathsf{H}_{m^*}^{\ell-1}$. With analogous checks one could verify that in DDH_3^0, $\mathcal{A}$ simulates perfectly $\mathsf{H}_{m^*}^{\ell}$. In conclusion the advantage of $\mathcal{A}$ and $\mathcal{D}$ are equal which proves the claim.

$\underline{\mathsf{H}^L_{\kappa N} \equiv \mathsf{H}_1}$: Upon conditioning on the event that no $k_0 = 1_{\mathbb{G}_2}$ in the first experiment, the two distributions are equal. Using a union bound, since the given environment execute the protocol al most L times, it follows that the statistical distance of the two experiments is smaller that $L \Pr[k_0 = 1_{\mathbb{G}_2}] = Lq^{-1}$, that is negligible.

$\underline{\mathsf{H}_1 \equiv \mathsf{H}_2}$: The two experiment follow the same probability distribution, conditioning on $\mathbf{w}_{j,1}, \mathbf{w}_{j,2}$ for $j \in Q \setminus \{i\}$ because for $b \in \{1,2\}$, $\mathbf{w}_{i,b}, \bar{\mathbf{w}}_b \sim U(\mathbb{F}_q^4)$ and, calling $\mathbf{v}_b = -\sum_{j \neq i} \mathbf{w}_{j,b}$, the map

$$\varphi : \mathbb{F}_q^2 \times \mathbb{F}_q^2 \to \mathbb{F}_q^2 \times \mathbb{F}_q^2 \quad : \quad \varphi(\mathbf{x}, \mathbf{y}) = (\mathbf{x} + \mathbf{y}, -\mathbf{x} - \mathbf{v})$$

is bijective. Since applying φ to $\mathbf{w}_{i,1}$ and $\bar{\mathbf{w}}_1$ in the distribution generated in H_1 we get H_2 the two experiment have the same distribution.

Claim: *When $\mathcal{Z}$ is executed in H_3 then in the ℓ-th execution of the setup up to negligible probability the keys $\mathsf{sk}_{i,m}$ returned by P_i for $i \in [N] \setminus M$ are of the form*

$$\mathsf{sk}_{i,m} = \left(\left[\rho_m \cdot \left(\sum_{j \in Q} \mathbf{w}_{j,1} + m \sum_{j \in Q} \mathbf{w}_{j,2} \right) + \rho_m \cdot (\bar{\mathbf{w}}_1 + m \bar{\mathbf{w}}_2) \right]_2, [\rho_m]_2 \right)$$

for some $\rho_m \in \mathbb{F}_q$ where $\mathbf{w}_{j,b}$ are the values received by $\mathcal{F}^{\mathsf{Lin}}_{\mathsf{Czk}}$ and $(\bar{\mathbf{w}}_1, \bar{\mathbf{w}}_2)$ are the values returned by $\mathcal{F}^{4,\mathbb{F}_q}_{\mathsf{CT}}$.

We build an adversary $\mathcal{B}$ using $\mathcal{Z}$ that breaks DDH over $\mathbb{G}_1$ with advantage proportional to the probability that the event above does not happen in some execution.
Description of $\mathcal{B}$: Wait for the challenger to send $(\widetilde{\mathbf{k}}, \widetilde{\mathbf{h}}) \in \mathbb{G}_1^4$ and run $M \xleftarrow{\$} \mathcal{Z}$. Simulate the initial key generation and $\mathcal{F}_{\mathsf{VSS}}$ correctly, initialise $\mathsf{scnt} \leftarrow 0$, sample $r \xleftarrow{\$} \mathbb{F}_q$ and compute $\mathbf{k} \leftarrow \widetilde{\mathbf{k}}^r$. Upon receiving:

1. $(\mathsf{setup}, \mathsf{sid})$ from all honest users: Add $\mathsf{scnt} \leftarrow \mathsf{scnt}+1$. If $\mathsf{scnt} \neq \ell$ simulate correctly H_2. Otherwise let $(\mathsf{tossed}, sid, Q, R)$ be the message sent by $\mathcal{F}^{\lambda,D,1}_{\mathsf{CT}}$. Program $\mathcal{H}(R) = \mathbf{k}, k_0, d_m$ for $k_0 \xleftarrow{\$} \mathbb{G}_1 \setminus \{1_{\mathbb{G}_1}\}$ and $d_m \xleftarrow{\$} \mathbb{G}_2$

1.1. $(\mathsf{commit}, sid, \ldots), (\mathsf{prove}, sid, \mathbf{k}, (k_{j,1}, k_{j,2}), (\mathbf{w}_{j,1}, \mathbf{w}_{j,2}))$ or error from P_j for all $j \in Q \cap M$: Execute uncorrupted parties as in H_2.

1.2. (open, sid) for $\mathcal{F}_{\mathsf{Com}}$, $\mathcal{F}^{\mathsf{Lin}}_{\mathsf{Czk}}$ or error from P_j for all $j \in Q \cap M$: Keep executing parties honestly and let $(\mathsf{tossed}, sid, \bar{\mathbf{w}}_1, \bar{\mathbf{w}}_2)$ be the message sent by $\mathcal{F}^{4,\mathbb{F}_q}_{\mathsf{CT}}$

When at the end of the ℓ-th setup P_i returns $\mathsf{sk}_i = (\mathsf{sk}_{i,m})$ parse $\mathsf{sk}_{i,m} = (\mathbf{d}'_m, d'_m)$. If $r = 0$ return a random bit. Conversely, if $r \neq 0$ and for some $m \in \phi^{-1}([N] \setminus M)$

$$\mathbf{t}_m \;=\; \mathbf{d}'_m \cdot \left[\sum_{j \in Q} \mathbf{w}_{j,1} + m \sum_{j \in Q} \mathbf{w}_{j,2} + (\bar{\mathbf{w}}_1 + m \bar{\mathbf{w}}_2) \right]^{-1}_{d'_m} \neq (1_{\mathbb{G}_2}, 1_{\mathbb{G}_2})$$

then return 1 if $e(\widetilde{\mathbf{h}}, \mathbf{t}_m) = 1_{\mathbb{G}_T}$, 0 otherwise. Return a random bit if the above condition is not satisfied for any m.
Proof of Claim: When $\mathsf{scnt} \neq \ell$, $\mathcal{B}$ correctly simulates H_2 by construction. The same also holds when $\mathsf{scnt} = \ell$ because $\widetilde{\mathbf{k}} = (g_2, \widetilde{k}_2)$ with $\widetilde{k}_2 \sim U(\mathbb{G}_1)$, therefore $\mathbf{k} = (g_2^r, \widetilde{k}_2^r) \sim U(\mathbb{G}_1^2)$.

Next we show that for all $m \in \phi^{-1}([N] \setminus M)$, $e(\mathbf{k}, \mathbf{t}_m) = 1_{\mathbb{G}_T}$. This holds because if at the end of the setup $j \in Q$, with $j \neq i$ it means that all player sent $(\mathsf{accept}, sid, j)$ and therefore

$$e(\mathbf{k}, \mathbf{d}_{j,m}) \ = \ e(k_{j,1}k_{j,2}^m, d_m) \ = \ e(\mathbf{k}^{\mathbf{w}_{j,1}+m\mathbf{w}_{j,2}}, d_m) \ = \ e(\mathbf{k}, [\mathbf{w}_{1,j} + m\mathbf{w}_{2,j}]_{d_m}).$$

Recalling that $\mathbf{d}'_m$ is obtained randomising with σ_m the product of all $\mathbf{d}_{j,m}$ for $j \in Q$ and $[\bar{\mathbf{w}}_1 + m\bar{\mathbf{w}}_2]_{d_m}$ we conclude that

$$\begin{aligned} e(\mathbf{k}, \mathbf{d}'_m) \ &= \ e\left(\mathbf{k}, [\bar{\mathbf{w}}_1 + m\bar{\mathbf{w}}_2]_{d_m}\right)^{\sigma_m} \prod\nolimits_{j \in Q} e\left(\mathbf{k}, [\mathbf{w}_{j,1} + m\mathbf{w}_{j,2}]_{d_m}\right)^{\sigma_m} \ = \\ &= \ e\left(\mathbf{k}, [\bar{\mathbf{w}}_1 + m\bar{\mathbf{w}}_2]_{d'_m}\right) \prod\nolimits_{j \in Q} e\left(\mathbf{k}, [\mathbf{w}_{j,1} + m\mathbf{w}_{j,2}]_{d'_m}\right) \ = \\ &= \ e\left(\mathbf{k}, \left[(\bar{\mathbf{w}}_1 + m\bar{\mathbf{w}}_2) + \sum\nolimits_{j \in Q} \mathbf{w}_{j,1} + m \sum\nolimits_{j \in Q} \mathbf{w}_{j,2}\right]_{d'_m}\right). \end{aligned}$$

Taking the inverse it follows that $e(\mathbf{k}, \mathbf{t}_m) = 1_{\mathbb{G}_T}$. In conclusion call E the event that $\mathbf{t}_m$ is different from the identity for some m. If $\mathcal{B}$ is executed in DDH^1 then there exists x such that $\widetilde{\mathbf{h}} = \widetilde{\mathbf{k}}^x = \mathbf{k}^{xk^{-1}}$ where $k \neq 0$ up to negligible probability. Therefore $e(\widetilde{\mathbf{h}}, \mathbf{t}_m) = e(\mathbf{k}, \mathbf{t}_m)^{xr^{-1}} = 1_{\mathbb{G}_T}$ and $\mathcal{B}$ always returns 1.
Conversely if $\mathcal{B}$ is executed in DDH^0, $\widetilde{\mathbf{h}} \sim U(\mathbb{G}_2^2)$ but for $\mathbf{t}_m \neq [\mathbf{0}]_2$, the space $\mathcal{L} = \{\mathbf{u} : e(\mathbf{u}, \mathbf{t}) = [0]_T\}$ is a proper space of dimension 1 - if we think $\mathbb{G}_2^2$ as an $\mathbb{F}_q$-vector space. It follows that

$$\Pr\left[e(\widetilde{\mathbf{h}}, \mathbf{t}_m) = [0]_T\right] \ = \ \frac{|\mathcal{L}|}{|\mathbb{G}_2^2|} \ = \ \frac{q}{q^2} \ = \ \frac{1}{q}.$$

To sum up, calling $\beta \in \{0, 1\}$ such that $\mathcal{B}$ is executed in DDH^β and recalling that b is $\mathcal{B}$'s output we proved so far that

$$\Pr[b = 1 | \beta = 0, \ E, \ r \neq 0] = 1, \quad \Pr[b = 1 | \beta = 1, \ E, \ r \neq 0] = \frac{1}{q}.$$

In conclusion we evaluate the advantage of $\mathcal{B}$ as $\mathsf{Adv}(\mathcal{B}) \ =$

$$\begin{aligned} &= \ \Big| \Pr[b = 1 | \beta = 0] - \Pr[b = 1 | \beta = 1] \Big| \\ &= \ \Big| \Pr[b = 1 | \beta = 0, \ \neg E] \Pr[\neg E] + \Pr[b = 1 | \beta = 0, \ E] \Pr[E] - \\ &\quad - \Pr[b = 1 | \beta = 1, \ \neg E] \Pr[\neg E] - \Pr[b = 1 | \beta = 1, \ E] \Pr[E] \Big| \\ &= \ \Big| \Pr[b = 1 | \beta = 0, \ E] - \Pr[b = 1 | \beta = 1, \ E] \Big| \Pr[E] \\ &= \ \Big| \Big[\Pr[b = 1 | \beta = 0, \ E, \ r = 0] - \Pr[b = 1 | \beta = 1, \ E, \ r = 0] \Big] \Pr[r = 0, \ E] \\ &\quad + \ \Big[\Pr[b = 1 | \beta = 0, \ E, \ r \neq 0] - \Pr[b = 1 | \beta = 1, \ E, \ r \neq 0] \Big] \Pr[r \neq 0, \ E] \Big| \\ &= \ \left| \left(1 - \frac{1}{q}\right) \Pr[r \neq 0 | E] \right| \Pr[E] \\ &= \frac{q-1}{q} \Pr[r \neq 0, \ E] \ \geq \ \frac{q-1}{q} (\Pr[E] + \Pr[r \neq 0] - 1) \\ &= \frac{q-1}{q} \left(\Pr[E] - \frac{1}{q}\right). \end{aligned}$$

where the third equation follows as $\mathcal{B}$ returns a random bit when E does not hold and the fifth because $\mathcal{B}$ returns random bit when $r = 0$. If we let ε be a negligible function bounding the advantage of $\mathcal{B}$ then the claim is proven because

$$\frac{q\varepsilon}{q-1} + \frac{1}{q} \geq \Pr\left[E\right].$$

$\underline{\mathsf{H}_2 \equiv \mathcal{S} \circ \mathcal{F}_{\mathsf{Setup}}}$: follows by inspection.