Efficient and Universally Composable Single Secret Leader Election from Pairings

Dario Catalano¹, Dario Fiore², 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 put forward new SSLE solutions that advance the state of the art both from a theoretical and a practical front. On the theoretical side we propose a new definition of SSLE in the universal composability framework. We believe this to be the right way to model security in 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 give a concrete PEKS scheme with efficient distributed algorithms for key generation and encryption and that allows us to efficiently instantiate our abstract SSLE construction.

Our resulting SSLE protocol is very efficient, does not require participants to store any state information besides their secret keys and guarantees so called *on-chain efficiency*: the information to verify an election in the new block should be of size at most logarithmic in the number of participants. To the best of our knowledge, this is the first SSLE scheme achieving this property along with practical efficiency.

Table of Contents

1	Introduction1.1Our contribution1.2Other related work1.3Organization	3 4 7 8
2	Preliminaries2.1Notation2.2SXDH assumption2.3Functional Encryption2.4Functional Encryption for Modular Keyword Search2.5Our Realization of FE for Modular Keyword Search2.6Our Construction of FE for orthogonality2.7Non Interactive Zero-Knowledge2.8UC model and Ideal Functionalities2.9Single Secret Leader Election	8 8 9 9 10 11 11 12 12
3	Universally Composable SSLE 3.1 A parametrised definition	14 17
4	UC-secure SSLE from FE for Modular Keyword Search	17
5	An Efficient UC-secure SSLE from SXDH5.1 Intuition5.2 SSLE protocol with Ideal Setup Functionality5.3 Realising the Setup	18 18 20 21
6	Efficiency considerations	24
A	Statistical Distance	29
в	Sigma ProtocolsB.1 The Decryption RelationB.2 Linear relation in the exponent	29 29 31
С	Comparing UC and Game-Based DefinitionC.1 Proof of Proposition 2C.2 SSLE schemes from Parametrised SSLE	35 35 36
D	Postponed ProofsD.1 Selective Security of FE for OrthogonalityD.2 Proof of Theorem 1D.3 Proof of Theorem 2D.4 Proof of Theorem 3	38 38 40 44 57

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 of 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. In principle this problem could be solved using general multiparty computation. What makes such an approach problematic are however the efficiency requirements that are desired in a blockchain context. In particular, beyond being computationally efficient, an SSLE protocol should guarantee low communication complexity (i.e. the total number of exchanged messages should scale with O(N) or better, where N is the number of miners/users), and more importantly it should be *on-chain efficient*: the amount of bits to store on chain, per new block, should be small (ideally logarithmic in N).

The question of finding such an election protocol 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* (at most 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 to achieve the lowest communication complexity and on-chain efficiency; indeed 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 round of communication, 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. However, other aspects of this solution makes it far from practical. First, it is not on-chain efficient:

to make the election verifiable, O(t) bits of information must be stored in the new block (unless one applies a transformation through a general-purpose SNARK proof that t valid partial decryptions exist). Second, 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 to communicate and store in the new block a freshly shuffled list of N Diffie-Hellman pairs⁴ (along with a NIZK of shuffle). Notice that this makes the solution inherently not on-chain efficient. 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 also that both the iO and TFHE-based SSLE protocols 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 that is on-chain efficient and 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. In particular, we give a generic protocol based on public key encryption with keyword search (PEKS) [BDOP04], and then propose an efficient instantiation of it based on pairings under the SXDH assumption. The latter is our main technical contribution: it is a protocol that achieves the same (asymptotic) communication complexity as the TFHE-based solution from [BEHG20] while achieving, in addition, on-chain efficiency and much better practical performances. We refer to Table 1 for a comparison between ours and the previous solutions and to the next section for an overview of our protocol. We note that, although our protocol requires a total of 2 rounds of communication to prepare an election, the first round can be actually executed in a preprocessing phase and shared to prepare many elections, thus making the online rounds effectively 1, as in the other solutions. Moreover, the protocol *does not* require parties to keep any state across rounds of communication, besides their secret keys.

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.

SSLE	Security	Setup	Election efficiency		
	model		Rounds	Comm.	On-chain
iO	Game-based	trusted	0	O(1)	O(1)
TFHE	Game-based	trusted	1	O(t)	O(t)
Shuffle-N	Game-based	—	1	O(N)	O(N)
Shuffle- \sqrt{N}	Game-based	-	1	$O(\sqrt{N})$	$O(\sqrt{N})$
Ours	UC	trusted	1 + 1	O(t)	$O(\kappa \log N)$

Table 1. Comparison between the SSLE solutions from [BEHG20] and the SSLE of this work. 'On-chain' refers to the amount of information to be stored on chain in the new block after every election. Shuffle- \sqrt{N} achieves a weak unpredictability notion. Everywhere, in $O(\cdot)$ we include the fixed security parameter λ . κ is a statistical security parameter that gives meaningful security for $\kappa = \log N$.

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 following simple idea. For every election a small subset of users generates a ciphertext c that encrypts a random keyword $j \in \{0, \ldots, N-1\}$. At registration time, each 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 two phases: (1) a setup (done rarely) 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 procedure in which a randomly sampled committee of κ players generates a commitment to the election's ciphertext in a distributed way. The commitment is then opened in a distributed way. Whoever knows a secret key that decrypts the ciphertext is the leader.

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 very efficient (distributed) protocols for the setup and election phases. To devise such a PEKS we build on (a modified variant of) the functional encryption for orthogonality (OFE) scheme recently proposed by Wee [Wee17]. Furthermore we extend this functionality to test keywords equality mod N albeit the message space is over a large field \mathbb{F}_q . We refer to this new primitive as modular PEKS.

Informally, the committed ciphertexts created in the election procedure are (plain) El Gamal encryptions of Wee's ciphertexts. An immediate advantage of this approach is that it allows for a very efficient setup procedure: it merely consists in a threshold key generation for El Gamal followed by the key generation for the functional encryption scheme. When relying on a publicly available random beacon, we show that the latter can be realized efficiently in two rounds of communication, one of which only used to perform complaints.

More interestingly, however, our proposed scheme allows to complete step (2) efficiently both in terms of computation and communication. Indeed, our protocol manages to distributively create valid (committed) ciphertexts c (encrypting messages uniformly distributed in a given range) in one single round of communication! Moreover, this round of communication can be used to generate, in parallel, as many ciphertexts as one wishes, one for every future election. This way, the communication needed to perform an election effectively consists of only one round of communication in which O(t) parties send their partial opening of the election's ciphertext.

We note that the naïve approach of posting all these O(t) partial openings in the blockchain would destroy our claimed on-chain efficiency guarantees. Interestingly, we can do better than this. Parties can exchange the O(t) partial openings off-chain and store in the blockchain only much shorter aggregate values that still enable anyone to verify the correctness of the election. Recall that opening our committed ciphertexts consists in, distributively, decrypting corresponding El Gamal ciphertexts. Simplifying things a bit, in our case this is achieved by letting players exchange partial decryption shares $(K_{1,i}, K_{2,i})$ together with corresponding NIZKs. These shares are then (locally) multiplied together to get values (K_1, K_2) that can be used to retrieve the encrypted ciphertext c. Whoever is able to decrypt c correctly can then claim victory. Concretely, in our protocol, a user can claim victory by posting on the blockchain only (K_1, K_2) , together with a proof that she can correctly decrypt c. Surprisingly, we show that a potentially expensive aggregated NIZK proving correctness of (K_1, K_2) is not needed for our protocol to be secure, as we prove that coming up with different (K'_1, K'_2) which open the ElGamal commitment to another $c' \neq c$ that an adversary is able to decrypt, implies breaking the underlying functional encryption scheme.

Concrete efficiency and comparison to previous solutions. To confirm the concrete performances of our protocol we measure them for $N = 2^{14}$ users, as suggested in [Lab19]. Our results show that the communication costs of an election are 36.8 KB to generate the committed election's ciphertext, 2.9 MB for the partial decryptions, and less than 1 KB to claim victory. Importantly, out of all this information, only 37.8 KB per election have to be stored on-chain for verifiability.

The major cost in our protocol is that of setup, which for 2^{14} users would amount to 252 MB. This setup, however, is supposed to be performed rarely⁵. Indeed, in our protocol we can add new users to the system without running a full setup: they engage in a registration procedure that allows them to receive their secret keys, without altering the key material of other users. This can be done with only 73 KB of communication per registration. If we compare to the shuffle-N solution of Boneh et al. [BEHG20], our protocol can easily amortize the expensive setup and results in less communication. In the shuffle-N solution, the issue is that every time a new user is added (which always includes the winner of the previous election if he still wants to run) a new shuffle has to be communicated and posted on-chain: this is about 1 MB per shuffle for 2^{14} users. Concretely, if we assume 50^6 new users join before every election, after 100 elections the shuffle-N scheme generates 6.2 GB to be communicated and stored on-chain, whereas our protocol involves 1.8 GB of off-chain communication and only 5.3 MB of on-chain storage.

Our election protocol more in detail. At the heart of our protocol there is a very efficient method to generate committed ciphertexts of the form discussed above. Here we informally highlight the main ideas underlying this construction. Recall that we build our PEKS from a tailored variant of the functional encryption for orthogonality (OFE) scheme recently proposed by Wee [Wee17]. In OFE a ciphertext is associated to a vector \mathbf{x} , a secret key corresponds to a vector \mathbf{y} and decryption allows one to learn if $\mathbf{y}^{\top}\mathbf{x} = 0$. The basic idea of our (modular) PEKS from OFE is inspired to that of [KSW08] with a novel tweak.

⁵ As in the TFHE solution, our protocol in practice requires periodic setup to refresh the secrets shared when many new users join

⁶ This number is justified by [Lab19], where $O(\log^2 N)$ new users are expected

In what follows, to keep the presentation intuitive and simple, we present a simplified version of our methods that, in particular, supports vectors of dimension 2 (rather than 3 as in our actual scheme) and only allows to test equality of keywords (rather than equality mod N).

During setup, each party P_i receives a public and secret key mpk, sk_i of the OFE scheme, where sk_i is associated to the vector (1, i). If there were a magic way to directly produce an encryption c of (m, -1) such that m is uniform over [N] (and no user gains any extra information on m), then, using FE.Dec, each party could test if m = i by simply checking whether $(m, -1)^{\top}(1, i) = 0$. Clearly the only user able to do this could then claim victory. Unfortunately, since no such wizardry is currently known, we go for the next best option: we develop a very fast, one round protocol to jointly produce a *commitment* of such a c^7 . The commitment is just a (standard) El Gamal encryption of c that can be (distributively) opened in one round of communication.

In this informal presentation, we explain how to generate the (committed) ciphertext, in the simpler case where m is allowed to lie in the slightly larger interval $[\kappa N]$. Our underlying ciphertexts have the following shape

$$\mathbf{c}_0 = [s\mathbf{a}]_1, \quad c_1 = \begin{bmatrix} m\sigma + s\mathbf{a}^\top \mathbf{w}_1 \end{bmatrix}_1, \quad c_2 = \begin{bmatrix} -\sigma + s\mathbf{a}^\top \mathbf{w}_2 \end{bmatrix}_1.$$

where $[\mathbf{a}]_1, [\mathbf{a}^\top \mathbf{w}_1]_1, [\mathbf{a}^\top \mathbf{w}_2]_1$ are public key elements and s, σ are random values. Note that in Wee's scheme $\sigma = s\mathbf{a}^\top \mathbf{u}$, with $[\mathbf{a}^\top \mathbf{u}]_1$ being an extra component of the public key. Using the random beacon, we begin by generating a (small) election committee $Q \subseteq [N]$ of size κ and two (random) group elements G, H that can be interpreted as an ElGamal encryption of $[\sigma]_1$ in the following way⁸

$$G = g^{\theta}, \qquad H = h^{\theta} [\sigma]_1$$

where (g, h) is the El Gamal public key and θ, σ are random and unknown to participants. Using this public information, each player $P_i \in Q$ can create (committed) encryptions of m_i by simply choosing random r_i, ρ_i, s_i , and $m_i \in [N]$ and broadcasting $[s_i \mathbf{a}]_1$ together with

$$G^{m_{i}} \cdot g^{r_{i}} = g^{\theta m_{i} + r_{i}} \qquad H^{m_{i}} \cdot h^{r_{i}} \cdot \left[s_{i} \mathbf{a}^{\top} \mathbf{w}_{1}\right]_{1} = h^{\theta m_{i} + r_{i}} \left[m_{i} \sigma + s_{i} \mathbf{a}^{\top} \mathbf{w}_{1}\right]_{1}$$
$$G^{-1} \cdot g^{\rho_{i}} = g^{\rho_{i} - \theta} \qquad H^{-1} \cdot h^{\rho_{i}} \cdot \left[s_{i} \mathbf{a}^{\top} \mathbf{w}_{2}\right]_{1} = h^{\rho_{i} - \theta} \left[-\sigma + s_{i} \mathbf{a}^{\top} \mathbf{w}_{2}\right]_{1}$$

All these (committed) ciphertexts share the same randomness σ and can thus be multiplied together to produce the final (committed) ciphertext of the vector ($m = \sum_{i \in Q} m_i, -1$). Note that the message m lies in the larger interval $[\kappa N]$ but $m \mod N$ is uniform over [N] as long as so is at least one of the m_i 's. Finally, as mentioned earlier, our actual realization (cf. Section 2.5) works around this issue by managing to test equalities modulo N.

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]

⁷ We stress here that no efficient single round solution to directly produce c seems possible because of rushing attacks.

 $^{^{8}}$ For clarity note that group operations are denoted multiplicatively, and that we make use of the bracket notation, cf. Section 2.1

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.

1.3 Organization

In the next section we start by introducing the notation, the computational assumptions and the cryptographic primitives used by our schemes. There we also recall the game-based definition of SSLE from [BEHG20]. Next, in section 3 we give our definition of SSLE in the universal composability framework, and in section 4 we propose our generic SSLE construction from PEKS. Section 5 includes our main contribution, that is our efficient SSLE protocol from the SXDH assumption. Finally, in section 6 we discuss the efficiency of our protocol in a realistic scenario and compare it with the SSLE based on shuffles by Boneh et al. [BEHG20].

2 Preliminaries

2.1 Notation

 $\lambda \in \mathbb{N}$ denotes the security parameter. A function $\varepsilon(\lambda)$ is said *negligible in* λ if it vanishes faster than the inverse of any polynomial in λ . $[n] = \{0, \ldots, n-1\}$. Bold font $(\mathbf{a}, \mathbf{u}, \mathbf{w}, \ldots)$ denotes vectors with entries in a given field or a group. $x \leftarrow^{\$} S$ 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 $\mathcal{G}(\lambda)$ a bilinear group generator, that is an algorithm which returns the description of a bilinear group $\mathbf{bg} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$, 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 $g_T = e(g_1, g_2)$ as 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] where no efficiently computable isomorphism between \mathbb{G}_1 and \mathbb{G}_2 is known.

 \mathbb{F}_q is 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 $\mathbf{g}^{\mathbf{a}} = g_1^{a_1} \cdots g_n^{a_n}$.

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 $bg = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2) \leftarrow^{\$} \mathcal{G}(1^{\lambda}).$

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 DDH⁰ a game in which \mathcal{A} received the first distribution and DDH¹ 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 two tuples

 $([a_1]_s, \dots, [a_n]_s, [b]_s, [c_1]_s, \dots, [c_n]_s) \qquad ([a_1]_s, \dots, [a_n]_s, [b]_s, [a_1b]_s, \dots, [a_nb]_s),$

denoted respectively as DDH_n^0 and DDH_n^1 , for random $a_i, b, c_i \sim U(\mathbb{F}_q)$. 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 (FE.Setup, FE.Enc, FE.KeyGen, FE.Dec) of PPT algorithms such that

- FE.Setup $(1^{\lambda})^{\$} \rightarrow (\mathsf{mpk}, \mathsf{msk})$ generates the secret and public master keys.
- FE.Enc $(m, mpk; r) \rightarrow c$ returns a ciphertext. Randomness r may be omitted.
- FE.KeyGen $(f, \mathsf{msk})^{\$} \rightarrow \mathsf{sk}_f$ returns a key associated to the function $f \in \mathsf{F}$.
- 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 mpk, msk \leftarrow ^{\$} FE.Setup (1^{λ}) , $c \leftarrow$ ^{\$} FE.Enc (m, mpk) , sk_f \leftarrow ^{\$} FE.KeyGen (f, msk) , then up to negligible probability FE.Dec $(c, f, \mathsf{mpk}, \mathsf{sk}_f) = f(m)$.

We recall the notion of selectively secure FE, which suffices for our goals.

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

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

2.4 Functional Encryption for Modular Keyword Search

Recall that the keyword search functionality [BDOP04, ABC⁺05] is defined as $\mathsf{F}_{\mathsf{ks}} = \{f_y : \mathcal{X} \to \{0,1\}\}$, where each function $f_y \in \mathsf{F}_{\mathsf{ks}}$ labelled by $y \in \mathcal{X}$ is such that $f_y(x)$ returns 1 if x = y and 0 otherwise. Our realization works with a generalisation of the above where equality are checked modulo a given integer. Formally we call modular keyword search functionality $\mathsf{F}_{\mathsf{mks}}^{\kappa} = \{f_y : \mathbb{F}_q \times \mathbb{F}_q \to \{0,1\}\}$ parametrized by a positive integer κ of polynomial size, where each function f_y labelled by $y \in \mathbb{F}_q$ are such that $f_y(x, n)$ returns 1 if $x = y + \delta n$ for some $\delta \in [\kappa]$, and 0 otherwise. Observe that when $y \in [n]$ and $x \in [\kappa n]$, then $f_y(x, n) = 1$ if and only if $x = y \mod n$.

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

- 1: $m_0, m_1 \leftarrow^{\$} \mathcal{A}$
- 2: Sample $b \leftarrow^{\$} \{0,1\}, \mathsf{mpk}, \mathsf{msk} \leftarrow^{\$} \mathsf{FE}.\mathsf{Setup}(1^{\lambda}), c \leftarrow^{\$} \mathsf{FE}.\mathsf{Enc}(m_b, \mathsf{mpk})$
- 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 $\mathsf{sk}_j \leftarrow^{\$} \mathsf{FE}.\mathsf{KeyGen}(f,\mathsf{msk})$ and send $\mathcal{A} \leftarrow \mathsf{sk}_f$
- 6: When $\mathcal{A} \to b'$: Return b == b'

2.5 Our Realization of FE for Modular Keyword Search

We realize our FE scheme for the keyword search functionality $\mathsf{F}_{\mathsf{mks}}^{\kappa}$ 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 $f_{\mathbf{y}}$, defined by a vector $\mathbf{y} \in \mathbb{F}_q^n$, 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 and OFE scheme already appears in previous work [KSW08]. In this paper, we tweak this construction in order to support $\mathsf{F}_{\mathsf{mks}}^{\kappa}$ described earlier (see Fig. 1). The idea is that $m = \gamma + \delta n$ if and only if $(m, -1, -n)^{\top}(\gamma, 1, \delta)$ for some $\delta \in [\kappa]$. Therefore, using an OFE scheme with dimension 3, a ciphertext for m and n is an encryption of the vector $\mathbf{x}_{m,n} = (m, -1, -n)$, while a key for γ is a collection of keys for the vectors $\mathbf{y}_{\gamma,\delta} = (1, \gamma, \delta)$, with $\delta \in [\kappa]$. This way, decryption can be realized by testing if one of the keys successfully decrypts.

$MKS.Setup(1^{\lambda}):$	$MKS.Enc(m,n,mpk){:}$
$ \begin{array}{ c }\hline (mpk',msk') \leftarrow^{\$} FE.Setup(1^{\lambda},3) \\ \hline \\ \mathbf{Return} \ mpk',msk' \end{array} $	$\mathbf{x}_{m,n} \leftarrow (m, -1, -n)$ Return $c \leftarrow$ [§] FE.Enc($\mathbf{x}_{m,n}, mpk$)
$MKS.KeyGen(\gamma,msk)\text{:}$	$MKS.Dec(c,y,mpk,sk_y):$
For $\delta \in [\kappa]$: $sk_{\gamma,\delta} \leftarrow^{\$} FE.KeyGen((1, y, \delta), msk)$	$ \overline{ \text{Set } \mathbf{y}_{\gamma,\delta} \leftarrow (1,\gamma,\delta) \text{ for all } \delta \in [\kappa] } $ $ \mathbf{If } \exists \delta \in [\kappa] : FE.Dec(c,\mathbf{y}_{\gamma,\delta},mpk,sk_{\gamma,\delta}) = 1 $
$\textbf{Return} \hspace{0.1 cm} sk_{\gamma} \leftarrow (sk_{\gamma,0}, \ldots, sk_{,\gamma,\kappa-1})$	Return 1. Else Return 0

Fig. 1. Our FE for $\mathsf{F}_{\mathsf{mks}}^{\kappa}$ from FE for orthogonality

Note however that the resulting construction is secure under a weaker notion in which the adversary, who initially query an encryption of (m_0, n_0) and (m_1, n_1) , can only asks secret keys for keywords γ such that $\gamma \neq m_0 + \delta n_0$ and $\gamma \neq m_1 + \delta n_1$ for all $\delta \in [\kappa]$. This restriction (often referred to as weak attribute-hiding) is sufficient in our application as we want to hide the winner's index $m \mod n$ only from those users that haven't won i.e. from those holding keys for $\gamma \neq m \mod n$.

$FE.Setup(1^{\lambda}, n)$:	$FE.KeyGen(\mathbf{y},msk)\text{:}$		
Sample $\mathbf{a}, \mathbf{w}_1, \dots, \mathbf{w}_n \leftarrow^{\$} \mathbb{F}_q^2$ $mpk \leftarrow ([\mathbf{a}]_1, [\mathbf{a}^\top \mathbf{w}_1]_1, \dots, [\mathbf{a}^\top \mathbf{w}_n]_1)$ $msk \leftarrow (\mathbf{w}_i)_{i=1}^n$ and return (mpk, msk)	$ \overline{r \leftarrow^{\$} \mathbb{F}_{q} \setminus \{0\} } $ Return sk _y $\leftarrow \left[\sum_{i=1}^{n} ry_{i} \mathbf{w}_{i}\right]_{2}, [r]_{2} $		
$FE.Dec(c, \mathbf{y}, mpk, sk_{\mathbf{y}})$:	$FE.Enc(\mathbf{x},mpk)\text{:}$		
Parse $c = (\mathbf{c}_0, c_i)_{i=1}^n$ with $\mathbf{c}_0 \in \mathbb{G}_1^2$ Parse $sk_{\mathbf{y}} = (\mathbf{d}_0, d_1)$ with $\mathbf{d}_0 \in \mathbb{G}_2^2$	$\sigma, s \leftarrow^{\$} \mathbb{F}_{q} \setminus \{0\}$ $c_{i} \leftarrow [\sigma x_{i} + s\mathbf{a}^{\top} \mathbf{w}_{i}]_{1}$ $\mathbf{P} \leftarrow (\mathbf{u} \in [\sigma \mathbf{x}_{i} + s\mathbf{a}^{\top} \mathbf{w}_{i}]_{1}$		
Return $e(\mathbf{c}_0, \mathbf{d}_0) \stackrel{?}{=} e(c_1^{y_1} \cdots c_n^{y_n}, d_1) \neq 1$	Return $c \leftarrow ([sa]_1, c_1, \ldots, c_n).$		

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

2.6 Our Construction of FE for orthogonality

We adapt to our setting the, pairing-based, FE for orthogonality proposed by Wee in [Wee17]. Our modified scheme is detailed in Figure 2. 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

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 (NIZK.G, NIZK.P, NIZK.V) where: NIZK.G generates a common reference string crs; NIZK.P(crs, x, w) given $(x, w) \in \mathcal{R}$, outputs a proof π ; NIZK.V(crs, x, π), 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 crs \leftarrow ^{\$} NIZK.G(1^{$\lambda$}) and all $(x, w) \in \mathcal{R}$, NIZK.V (crs, x, NIZK.P(crs, x, w)) = 1 holds with probability 1. In our protocols we require the NIZKs to satisfy the notions of weak simulation extractability [Sah99] and zero-knowledge [FLS90].

About the first property, 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 evaluations need to be salted with a unique session ID. Note that 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 about group elements. The first one checks whether two vectors $\mathbf{g}, \mathbf{h} \in \mathbb{G}_1^n$ are proportional, i.e., there exists $a \in \mathbb{F}_q$ s.t. $\mathbf{g}^a = \mathbf{h}$. The second one generalise the previous to linear maps. The third one asks for solutions to the linear system $A\mathbf{x} = \mathbf{b}$ where A, \mathbf{b} are given in the exponent and the last component x_n lies in a prescribed range. Formally

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

We also use \mathcal{R}_{Enc} and \mathcal{R}_{Dec} which relates to a given functional encryption scheme. The first one, given a ciphertext, requires knowledge of the message and randomness used to generate it. The

second one instead, given a tuple (mpk, c, f, x) asks for a *correct secret key* sk_f that decrypts c to x. Below we also introduce a language $\mathcal{L}_{\mathsf{key}}$ to formally capture the notion of correct secret key.

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

Notice that asymmetric encryption scheme, that are a special case of FE with only one function (the identity over the message space) are also captured by this definition abusing notation.

To construct our protocols, we assume the existence of a NIZK argument for each of these relations. We note that all of them can be proved through a sigma protocol, and that Fiat-Shamir based NIZKs from sigma protocols are weakly-simulation-extractable [FKMV12] based on a special property 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}_{DDH} and \mathcal{R}_{Lin} , we can use generalised Schnorr protocols provided in [Mau15]. For \mathcal{R}_{LR} we propose a variant of the folklore solution based on binary decomposition⁹, detailed in the appendix, Section B.2. We propose a sigma protocol for \mathcal{R}_{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 realises \mathcal{F} if it is indistinguishable from $\mathcal{F} \circ \mathcal{S}$ for a given PPT simulator \mathcal{S} . 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 two functionalities required in our construction: \mathcal{F}_{zk} and $\mathcal{F}_{CT}^{\mathcal{D}}$ which respectively models a zero-knowledge proof of knowledge and a random beacon. The first one was introduced in [CF01], even though our definition slightly differ as all the recipients receive the output messages. This deviation is justified by our assumption of an authenticated broadcast channel. \mathcal{F}_{CT} instead was recently introduced in [CD20] and realised assuming a suboptimal honest majority under standard assumptions.

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

⁹ In this case 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 ZK Functionality $\mathcal{F}_{zk}^{\mathcal{R}}$: Upon receiving (prove, sid, x, w) from P_i , with sid being used by P_i for the first time: if $(x, w) \in \mathcal{R}$, broadcast (proof, sid, i, x).

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

by three properties *uniqueness*, *fairness* and *unpredictability* respectively implying that 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. The second 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 defacto present in the security proofs of all schemes in [BEHG20]. The third 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 $\langle \vartheta(n) \rangle$ 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 five protocols (SSLE.Setup, SSLE.Reg, SSLE.Elect, SSLE.Claim, SSLE.Vrf) executed among N users, such that

- SSLE.Setup returns pp to every player and sp_i to P_i .
- SSLE.Reg_{pp}(i) registers player P_i for future elections
- SSLE.Elect_{pp} returns publicly a challenge c.
- SSLE.Claim_{pp} $(c, sp_i, i) \rightarrow \pi/\perp$ returns publicly a proof to claim victory.
- SSLE.Vrf_{pp} $(c, \pi, i) \rightarrow 0/1$ verifies the correctness of a claim.

An SSLE scheme is said to admit revocation if there is a sixth protocol SSLE. $\operatorname{Rev}_{pp}(i)$ which revokes the registration of player P_i from future elections

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

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

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 $\eta(\lambda)$ -fairness if for every PPTalgorithm \mathcal{A} that corrupts T < t players there exists a negligible function ε such that

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

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

Definition 8 (Unpredictability). An SSLE scheme satisfies (t, ϑ) -threshold $\eta(\lambda)$ -unpredictability if for every PPTA that corrupts T < t parties there exists a negligible function ε such that

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

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

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

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

- 1: When $M \leftarrow \mathcal{A}(1^{\lambda}, N)$, simulate P_i for $i \in [N] \setminus M$ interacting with \mathcal{A}
- $2: \quad \text{Execute SSLE.Setup} \to \mathsf{pp}, \mathsf{sp}_i \text{ for } i \in [N] \setminus M. \text{ Set } s \leftarrow 0, \, R \leftarrow \varnothing$
- 3: When register, $i \leftarrow \mathcal{A}$: Run SSLE. $\mathsf{Reg}_{\mathsf{pp}}(i)$ and add $R \leftarrow R \cup \{i\}$
- 4: When revoke, $i \leftarrow \mathcal{A}$: Run SSLE. $\mathsf{Rev}_{pp}(i)$ and remove $R \leftarrow R \setminus \{i\}$
- 5: When elect $\leftarrow \mathcal{A}$ and $|R \cap M| < \vartheta(|R|)$: Execute SSLE.Elect_{pp} $\rightarrow c_s$
- 6: $\pi_{i,s} \leftarrow \mathsf{SSLE.Claim}_{\mathsf{pp}}(c_s, \mathsf{sp}_i, i), \quad \mathcal{A} \leftarrow \pi_{i,s} \quad \forall i \in \mathbb{R} \setminus M$
- 7: $\pi_{i,s} \leftarrow \mathcal{A} \quad \forall i \in R \cap M; \quad s \leftarrow s+1$
- 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 SSLE.Vrf_{pp} $(c_t, \pi_{i,s'}, i) =$ SSLE.Vrf_{pp} $(c_t, \pi_{j,s'}, j) = 1$

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 $\mathcal{F}_{\mathsf{SSLE}}$.

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

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

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

- 1: When $M \leftarrow \mathcal{A}(1^{\lambda}, N)$, simulate P_i for $i \in [N] \setminus M$ interacting with \mathcal{A}
- 2: Execute SSLE.Setup \rightarrow pp, sp_i for $i \in [N] \setminus M$. Set $s \leftarrow 0, R \leftarrow \varnothing$.
- 3: When register, $i \leftarrow \mathcal{A}$: Run SSLE. $\mathsf{Reg}_{pp}(i)$ and add $R \leftarrow R \cup \{i\}$
- 4: When revoke, $i \leftarrow \mathcal{A}$: Run SSLE. $\operatorname{Reg}_{pp}(i)$ and remove $R \leftarrow R \setminus \{i\}$
- 5: When elect $\leftarrow \mathcal{A}$ and $|R \cap M| < \vartheta(|R|)$: Execute SSLE.Elect_{pp} $\rightarrow c_s$
- 6: $\pi_{i,s} \leftarrow \mathsf{SSLE.Claim}_{\mathsf{pp}}(c_s, \mathsf{sp}_i, i), \quad \mathcal{A} \leftarrow \pi_{i,s} \quad \forall i \in \mathbb{R} \setminus M$
- $7: \qquad \pi_{i,s} \leftarrow \mathcal{A} \quad \forall i \in R \cap M; \quad s \leftarrow s+1$
- 8: When chall $\leftarrow \mathcal{A}$ and $|R \cap M| < \vartheta(|R|)$: Call n = |R| and $\tau = |R \cap M|$;
- 9: Execute SSLE.Elect_{pp} $\rightarrow c_s$
- 10: $\pi_{i,s} \leftarrow \mathsf{SSLE.Claim}_{\mathsf{pp}}(c, \mathsf{sp}_i, i) \quad \forall i \in \mathbb{R} \setminus M; \quad j \leftarrow \mathcal{A}$
- 11: Return 1 if any protocol fails or if SSLE.Vrf_{pp} $(c_s, \pi_{j,s}, j) = 1, 0$ otherwise

At a high-level, $\mathcal{F}_{\mathsf{SSLE}}$ consists of the following commands. By using (register) a user can register to 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 (fake_rejected, *eid*, *j*) command is reserved to the adversary and makes ideal functionality announce to everyone that the (corrupted) user *j* is *not* the winner. This models 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.

Following the same idea that led us to the (t, ϑ) -threshold definitions in Section 2.9, we now specify a class of environments that (1) corrupts M with |M| < t parties; (2) induce \mathcal{F}_{SSLE} to perform elections only when $|R \cap M| < \vartheta(|R|)$.

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

The SSLE functionality *F*_{SSLE}: Initialise *E*, *R* ← Ø, *n* ← 0 and let *M* be the set of corrupted parties. Upon receiving:
(register) from *P_i*: add *R* ← *R* ∪ {(*i*, *n*)}, broadcast (registered, *i*) and set *n* ← *n* + 1.
(elect, *eid*) from all honest parties: if *R* ≠ Ø and *eid* was not requested before sample (*j*, γ) ←^{\$} *R* and send (outcome, *eid*, 1) to *P_j* and (outcome, *eid*, 0) to *P_i* for (*i*, ·) ∈ *R*, *i* ≠ *j*. Store *E* ← *E* ∪ {(*eid*, *j*)}.
(reveal, *eid*) from *P_i*: if (*eid*, *i*) ∈ *E* broadcast (result, *eid*, *i*). Otherwise broadcast (rejected, *eid*, *j*).
(fake_rejected, *eid*, *j*) from the adversary: if *P_j* is corrupted broadcast (rejected, *eid*, *j*).

a set M of parties with |M| < t and such that at each step, calling R the set of registered users, $|R \cap M| < \vartheta(|R|)$.

Definition 10. A (t, ϑ) -threshold statically secure UC-SSLE is a protocol Π that (t, ϑ) -securely realise \mathcal{F}_{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 one. 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

SSLE.Setup:	$SSLE.Reg_{pp}(i)$:			
$P_i \text{ sets } (pp, sp_i, eid) \leftarrow (\bot, \bot, 0)$	P_i sends (register, i) to Π Others wait (registered, i) $\leftarrow \Pi$			
$SSLE.Elect_{pp}$:				
All players send (elect, eid) to Π and update $eid \leftarrow eid + 1$ When P_i receives (outcome, $eid, \cdot) \leftarrow \Pi$: return eid				
$SSLE.Claim_{pp}(c, sp_i, i) \text{:}$	$SSLE.Vrf_{pp}(c,\pi,i) \text{:}$			
Send (reveal, c) to Π . Return $\pi \leftarrow \perp$	When $(result, c, i) \leftarrow \Pi$ return 1 When $(rejected, c, i) \leftarrow \Pi$ return 0			

Fig. 3. The derived SSLE scheme from a UC-SSLE protocol \varPi

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 taking one of the protocols from [BEHG20] (e.g., the one based on TFHE or the one based on Shuffling) they cannot be UC-secure if the zero-knowledge proofs they employ are not UC-secure.¹⁰ In [BEHG20], these protocols are proven secure without making any UC assumption

¹⁰ 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.

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 parametrised 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. In order to leverage security and efficiency we present here a parametrised functionality $\mathcal{F}_{SSLE}^{\kappa,\eta}$ that allows the adversary with probability smaller than $2^{-\kappa}$ to control a given election and which may not elect any user with probability smaller than $2^{-\eta}$.

> The Parametrised SSLE functionality *F*^{κ,η}_{SSLE}: Initialise *E*, *R* ← Ø, *n* ← 0 and let *M* be the set of corrupted parties. Upon receiving:
> (register) from *P_i*: add *R* ← *R* ∪ {(*i*, *n*)}, broadcast (registered, *i*) and set *n* ← *n* + 1.
> (elect, *eid*) from all honest parties: if *eid* was not requested before leak (electing, *eid*). Upon receiving (prob, *eid*, *p*₁, *p*₂) with *p*₁ ≤ 2^{-κ}, *p*₂ ≤ 2^{-η}: With probability *p*₁ leak (corrupted, *eid*) and wait for the adversary to reply with (infl, *eid*, *j*). Else, with probability *p*₂ set *j* ←⊥. If the previous action are not performed, sample (*j*, *n'*) ←[§] *R*. Send (outcome, *eid*, 1) to *P_j* and (outcome, *eid*, 0) to *P_i* for (*i*, ·) ∈ *R*, *i* ≠ *j*. Add *E* ← *E* ∪ {(*eid*, *j*)}.
> (reveal, *eid*) from *P_i*: if (*eid*, *i*) ∈ *E* broadcast (result, *eid*, *i*). Otherwise broadcast (rejected, *eid*, *i*).
> (fake_rejected, *eid*, *j*) from the adversary: if *P_j* is corrupted broadcast (rejected, *eid*, *j*).

Setting $\kappa = \eta = \Theta(\lambda)$ we get back a functionality equivalent to $\mathcal{F}_{\mathsf{SSLE}}$. However for smaller values of κ, η we can now capture schemes achieving weaker fairness and unpredictability, which in practice might be sufficient, especially if these lead to significant efficiency gains. In Section C.2 we show that applying the construction in Figure 3 to a protocol realising $\mathcal{F}_{\mathsf{SSLE}}^{\kappa,\eta}$ with less than $\vartheta(n)$ statically corrupted players, yields an SSLE scheme with $(2^{-\kappa} + 2^{-\eta})$ -fairness and $\xi(\kappa)$ -unpredictability with

$$\xi(\kappa) = \sup_{n \in \mathbb{N}} \left(\frac{n}{n - \vartheta(n)} \right) \cdot \frac{1}{2^{\kappa}} \cdot \frac{2^{\eta}}{2^{\eta} - 1}.$$

For fairness, the $2^{-\kappa} + 2^{-\eta}$ bound simply means that for $\kappa, \eta = \log N$ an adversary controlling T parties, wins the election with probability (T+2)/N. This is the same winning probability of an adversary that runs a fair election but corrupts two single extra players.

4 UC-secure SSLE from FE for Modular Keyword Search

We begin presenting a generic construction of a UC-SSLE protocol based on modular keyword search FE, which, besides being of independent interest, serves as a stepping stone toward our full construction. More specifically we realise $\mathcal{F}_{SSLE}^{\kappa,\eta}$ assuming there exists a protocol Π that securely distributes keys and on request produces ciphertexts encrypting messages uniformly distributed.

Our construction roughly works as follows: Initially the public key mpk is distributed among N user. To perform the *n*-th registration for P_i , parties run Π to give sk_n to P_i . Finally, when an election is requested, users generate with Π a challenge ciphertext c that encrypts a message m, n, with $m \in [\kappa n]$ such that $m \mod n \sim U([n])$, and 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.

Unfortunately, even if this solution can already be proven secure in the game-based definition, it is not UC-secure yet. The reason is that, if at a given round a ciphertext c encrypting m, nwith $m = \gamma + \delta n$ is returned, γ being associated to an honest user, the adversary could re-register malicious users until he gets sk_m and then test that $\mathsf{MKS.Dec}(c, m, \mathsf{mpk}, \mathsf{sk}_m) = 1$. This makes the protocol hard to simulate as the ciphertext produced needs to contain the winner's index – which the simulator cannot know in advance.

To prevent this issue we introduce a set S of forbidden keys: each time a user wins with key sk_{γ} , the indices $\gamma + \delta n$ for $\delta \in [\kappa]$ are added to S and, each time a new user joins, n is set to be the next integer not lying in S. However this introduce a probability $|[n] \cap S| \cdot n^{-1}$ to produce a ciphertext none can decrypt, meaning that none is elected. A way to keep this smaller than $2^{-\eta}$ is to perform a new setup every time it would exceed this bound.

To proceed we formally define a functionality \mathcal{F}_{SnC} , that shapes behaviour and security of Π , and a protocol $\{P_{\mathsf{MKS-SSLE}}^{(i)} : i \in [N]\}$ in the \mathcal{F}_{SnC} -hybrid model realising $\mathcal{F}_{\mathsf{SSLE}}^{\kappa,\eta}$. Security is proven in Appendix D.2.

The Setup and Challenge Functionality F_{SnC}:
Generate crs ←^{\$} NIZK.G_{Dec}(1^λ), mpk, msk ←^{\$} KS.Setup(1^λ). Set S ← Ø and n ← 0. Upon receiving:
(setup) from P_i: send (input, crs, mpk) to P_i
(ch_request, eid) from all honest parties: If eid was not requested before, sample m ←^{\$} [κn] and compute c ←^{\$} KS.Enc((m, n), mpk). Broadcast (challenge, eid, c). If m ∉ S, divide m = γ + δn with γ < n and add S ← S ∪ {γ + δ'n : δ' ∈ [κ]}.
(keygen) from P_i: When honest users agree, n ← n + 1 until n ∉ S, set sk_n ← MKS.KeyGen(n, msk) send (key, sk_n) to P_i and broadcast (key_request, i, n).

Theorem 1 The protocol $\{P_{\mathsf{MKS-SSLE}}^{(i)} : i \in [N]\}$ securely realises $\mathcal{F}_{\mathsf{SSLE}}^{\lambda,\eta}$ 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.

5.1 Intuition

The idea is to instantiate the generic construction of Section 4 with the modular KS scheme obtained applying the transformation in Fig. 1 to our OFE in Fig. 2.

Party $P_{\mathsf{MSK-SSLE}}^{(i)}$ realising $\mathcal{F}_{\mathsf{SSLE}}^{\lambda,\eta}$:

- Set $C, R, \mathcal{K} \leftarrow \emptyset$, send setup to \mathcal{F}_{SnC} and wait for (input, crs, mpk). Upon receiving:
 - register: send keygen to F_{SnC} and for it to reply with (key, sk_γ). Store K ← K ∪ {sk_γ}.
 A request to generate a new secret key from F_{SnC}: accepts if there is no election currently in progress.
 - (key_request, j, n) from \mathcal{F}_{SnC} : return (registered, j) and set $R \leftarrow R \cup \{(j, n)\}$.
- (elect, *eid*): send (ch_request) to \mathcal{F}_{SnC} . When it replies with (challenge, *eid*, *c*), if there exists $\mathsf{sk}_{\gamma} \in \mathcal{K}$ such that $1 \leftarrow \mathsf{MKS}.\mathsf{Dec}(c, \gamma, \mathsf{mpk}, \mathsf{sk}_{\gamma})$, return (outcome, *eid*, 1). Otherwise return (outcome, *eid*, 0). Add $C \leftarrow C \cup \{(c, eid)\}$.
- (reveal, *eid*): if $(eid, c) \in C$ and $1 \leftarrow \mathsf{KS.Dec}(c, i, \mathsf{mpk}, \mathsf{sk}_{\gamma})$ for some $\mathsf{sk}_{\gamma} \in \mathcal{K}$, prove $\pi \leftarrow^{\$} \mathsf{NIZK.P}_{\mathsf{Dec}}(\mathsf{mpk}, c, \gamma, \mathsf{sk}_{\gamma})$ and broadcast (claim, eid, π, γ). Otherwise broadcast (claim, eid, \bot).
 - (claim, eid, π, γ) from P_j : if $(j, \gamma) \in R$ and $1 \leftarrow \mathsf{NIZK.V}_{\mathsf{Dec}}(\mathsf{crs}, \mathsf{mpk}, c, \gamma, \pi)$ return (result, eid, j), otherwise (rejected, eid, j)

The main challenge is to efficiently generate ciphertexts in a distributed way. To address this, we select a random committee $Q \subseteq [N]$ and have each member P_j secretly sample a value $m_j \in [n]$ and jointly generate an encryption of $m = \sum_{j \in Q} m_j$. A downside is that now $m \in [|Q|n]$. For this reason, we use the FE for the functionality $\mathsf{F}_{\mathsf{mks}}^{\kappa}$ with $\kappa = |Q|$. This way, decryption still provides a good way to test if one wins, as with this functionality the holder of secret key for index γ learns if $m = \gamma \mod n$. Also, if at least one m_j is uniform over [n], so is $m \mod n$. Moreover, as $|Q| = \kappa$ is a small parameter, the decryption of our scheme in Fig. 1 remains 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:

$$\mathbf{c}_0 = [s\mathbf{a}]_1, \ c_1 = \left[\sigma m + s\mathbf{a}^\top \mathbf{w}_1\right]_1, \ c_2 = \left[-\sigma + s\mathbf{a}^\top \mathbf{w}_2\right]_1, \ c_3 = \left[-n\sigma + s\mathbf{a}^\top \mathbf{w}_3\right]_1$$

While \mathbf{c}_0, c_2, c_3 would be easy to generate alone, as they linearly depend on s, σ , in c_1 we need to compute a product $\sigma \cdot m$. Standard MPC techniques could solve this issue within a few rounds, however we opt for a solution that requires each user to only speak once. First, we sample two group elements G, H through the random beacon and interpret them as the ElGamal encryption, with respect to a previously generated public key g, h, of $[\sigma]_1$. Next each player P_i for $i \in Q$ samples $m_i \in [n], s_i \in [n]$, and, using the linearity of ElGamal, computes and randomise an encryption of

$$c_{1,i} = \begin{bmatrix} \sigma m_i + s_i \mathbf{a}^\top \mathbf{w}_1 \end{bmatrix}_1, \quad c_{2,i} = \begin{bmatrix} -\sigma + s_i \mathbf{a}^\top \mathbf{w}_2 \end{bmatrix}_1, \quad c_{3,i} = \begin{bmatrix} -n\sigma + s_i \mathbf{a}^\top \mathbf{w}_3 \end{bmatrix}_1$$

Finally he sends these encrypted values together with $\mathbf{c}_{0,i} = [s_i \mathbf{a}]_1$ and a NIZK.

At this point everyone can locally set \mathbf{c}_0 as the product of $\mathbf{c}_{0,i}$, while to get the other components we assume that the secret key x of the ElGamal public key, s.t. $g^x = h$, was previously t-shared among all users. This way those elements are retrieved through a threshold decryption. In conclusion we point out that, as in the general construction in Section 4, we have to maintain a set S of keys that cannot be generated in order to keep the protocol simulatable, incurring occasionally in elections without leaders.

Finally, to complete the protocol we have to 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

In Figure 4 we show a protocol that securely realizes the $\mathcal{F}_{SSLE}^{\kappa,\eta}$ ideal functionality. To this end we use the following building blocks:

- The FE scheme for orthogonality in Fig. 2, denoted FE.
- NIZKs for \mathcal{R}_{DDH} , \mathcal{R}_{LR} and \mathcal{R}_{Dec} . For readability, we suppress the crs from the inputs of the prover and verifier algorithm.
- A functionality $\mathcal{F}_{\mathsf{SK}}$ that distributes public and private keys of our OFE scheme, and *t*-share a threshold ElGamal secret key sending privately the share f(j) to P_j and publicly $k_j = g^{f(j)}$.
- A random beacon \mathcal{F}_{CT}^{ch} returning G, H, Q with $G, H \sim U(\mathbb{G}_1^2)$ and $Q \subseteq [N], |Q| = \ell$ such that the probability that Q is contained in the set of corrupted parties is smaller than $2^{-\kappa}$. Note that t < N/2 implies $\ell \leq \kappa$.

Each user maintain (or recover from the public state) four sets C, R, S, \mathcal{K} respectively containing previous challenges, currently registered users, forbidden keys and owned secret keys.

Elections begin by invoking \mathcal{F}_{CT}^{ch} which returns (G, H, Q). In steps 6-8 users in Q interpret $(G, H) = (g^{\theta}, h^{\theta} \cdot [\sigma]_1)$ with $\sigma \sim U(\mathbb{F}_q)$, sample $m_i \in [n], s_i \in \mathbb{F}_q$ and produce encrypted shares of the challenge components. Then they sample r_i and ρ_i to re-randomize these ciphertext. Interestingly we observe that using the same randomness for the last two components does not affects security.

Next, in steps 11-15 we let $Q_0 \subseteq Q$ be the set of users who replied with a correct NIZK. Observe that, calling s, r, ρ, m the sum of the respective shares over Q_0 , then $G_1 = g^{r+\theta m}$ and $G_2 = g^{\rho+\theta}$. In order to decrypt each user produces $K_{1,i}, K_{2,i}$ that will open to $h^{r+\theta m}$ and $h^{\rho+\theta}$.

In steps 16-20, users locally multiply the elements sent by the committee and reconstruct, interpolating at the exponent, $K_1 = h^{r+\theta m}$ and $K_2 = h^{\rho+\theta}$. Since

$$\begin{split} \prod_{\mu \in Q_0} c_{1,\mu} &= h^{r+\theta m} \left[m\sigma + s\mathbf{a}^{\top} \mathbf{w}_1 \right]_1 \qquad H^{-1} \prod_{\mu \in Q_0} c_{2,\mu} = h^{-\rho-\theta} \left[-\sigma + s\mathbf{a}^{\top} \mathbf{w}_2 \right]_1 \\ H^{-n} \prod_{\mu \in Q_0} c_{3,\mu} &= h^{-n(\rho-\theta)} \left[-n\sigma + s\mathbf{a}^{\top} \mathbf{w}_3 \right]_1 \end{split}$$

applying K_1, K_2 they finally obtain all the components of the challenge c. At the end of an election (lines 20-22) each user verify whether or not he won.

When a user wins and receives a reveal command, he claim victory by sending both the elements K_1, K_2 and a proof of knowledge for a secret key $\mathsf{sk}_{\mathbf{y}_{\gamma,j}}$. The first part being required as we don't want to store on chain the threshold decryption. Even though this may sound insecure as another user could come up with different K'_1, K'_2 that let him win, in the proof of security we show that being able to do so implies breaking the selective security of our OFE. See Section D.3, Claim 8 for more details.

Finally we deal with the remaining commands as in the generic construction in Section 4.

Theorem 2 The protocol in Fig. 4 (t, ϑ) -threshold securely realizes $\mathcal{F}_{\mathsf{SSLE}}^{\kappa,\eta}$ in the $(\mathcal{F}_{\mathsf{CT}}, \mathcal{F}_{\mathsf{SK}})$ -hybrid model under the SXDH assumption, for t = |N/2| and $\vartheta(n) = |n/2|$

The Setup and Key Generation Functionality \mathcal{F}_{SK} $\begin{array}{l} n \leftarrow 0, \ S \leftarrow \varnothing, \ g \leftarrow^{\$} \ \mathbb{G}_{1}, \ f \leftarrow^{\$} \ \mathbb{F}_{q}[x]_{< t}, \ \mathsf{mpk}, \mathsf{msk} \leftarrow^{\$} \ \mathsf{FE.Setup}(1^{\lambda}, 3). \ \mathrm{Fix} \\ h \leftarrow g^{f(-1)}, \ k_{j} \leftarrow g^{f(j)}, \ \mathsf{pp} \leftarrow (\mathsf{mpk}, g, h, (k_{j})_{j=0}^{N-1}), \ \mathrm{leak} \ (\mathsf{setup_leak}, \mathsf{pp}, f(j))_{j \in M} \end{array}$ and wait for $(\mathsf{setup_infl}, \mathbf{w}^*_{\alpha}, f^*)^3_{\alpha=1}$ from the adversary \mathcal{Z} . Calling $\mathsf{mpk} = (\mathbf{z}_0, z_\alpha)^3_{\alpha=1}$, $\mathsf{msk} = (\mathbf{w}_{\alpha})_{\alpha=1}^{3}$, set $\mathsf{mpk} \leftarrow (\mathbf{z}_0, z_\alpha \cdot \mathbf{z}_0^{\mathbf{w}_\alpha^*})_{\alpha=1}^3, \qquad \mathsf{msk} \leftarrow (\mathbf{w}_\alpha + \mathbf{w}_\alpha^*)_{\alpha=1}^3, \qquad f \leftarrow f + f^*$ and update **pp**. Upon receiving: - (setup) from P_j : Send (input, pp, f(j)) to P_j . - (update, γ , n) from honest players: $S \leftarrow S \cup \{\gamma + \delta n : \delta \in [\kappa]\}$. - (keygen) from P_j : Broadcast (key_request). while $n \in S$, increase n by 1. Set $\mathsf{sk}_{n,\delta} \leftarrow \mathsf{FE}.\mathsf{KeyGen}(\mathbf{y}_{n,\delta},\mathsf{msk})$, and send (key, $(\mathsf{sk}_{n,\delta})_{\delta=0}^{\kappa-1}$) to P_j . - (infl, $(\mathbf{w}_i^*)_{i=1}^3$) from the adversary \mathcal{Z} : For each key $\mathsf{sk} = (\mathbf{d}, d)$ sent to P_j associated to

a vector \mathbf{y} , compute $\mathsf{sk}' = (\mathbf{d} \cdot [y_1 \mathbf{w}_1^* + y_2 \mathbf{w}_2^* + y_3 \mathbf{w}_3^*]_d, d)$ and send (key_update, sk') to P_j . Update msk setting $\mathbf{w}_{\alpha} \leftarrow \mathbf{w}_{\alpha} + \mathbf{w}_{\alpha}^*$.

Realising the Setup 5.3

For what regards the setup and key generation, in Protocol 4 we used for simplicity a functionality \mathcal{F}_{SK} . Here we describe how to realise it.

First of all, in order to emulate private communication channels, not available in our model but necessary to distribute secret parameters, we use an IND-CPA encryption scheme (AE.Setup, AE.Enc, AE.Dec). Second, as our NIZKs are randomised sigma protocols compiled with Fiat-Shamir, they only need access to a random oracle and in particular there is no need to instantiate a crs. Next, we need to distribute the secret key of the Threshold ElGamal scheme. This is addressed by deploying standard techniques from verifiable secret sharing.

Finally we have to generate the public and secret keys of the FE scheme in Figure 2. To this aim, recall that

$$\mathsf{mpk} = [\mathbf{a}]_1, (\left[\mathbf{a}^\top \mathbf{w}_\alpha\right]_1)_{\alpha=1}^3, \qquad \mathsf{sk}_{\mathbf{y}_{\gamma,\delta}} = [r(\mathbf{w}_1 + \gamma \mathbf{w}_2 + \delta \mathbf{w}_3)]_2, [r]_2.$$

Fixing $[\mathbf{a}]_1$ and $[r]_2$, which can be generated through a random beacon, the remaining components of these keys depends linearly on \mathbf{w}_{α} . Therefore we can again select a random committee and let each member P_i sample $\mathbf{w}_{\alpha,i} \leftarrow \mathbb{F}_q^2$. At a high level to produce either mpk or a secret key, users provide shares of it, which are then locally multiplied. When reconstructing a secret key moreover the receiver checks the shares and complain if they are malformed.

More in detail in our construction we will use

- NIZKs for $\mathcal{R}_{\mathsf{Enc}}$, $\mathcal{R}_{\mathsf{Dec}}$ and the ideal functionality $\mathcal{F}_{\mathsf{zk}}^{\mathsf{Lin}}$. Two random beacon $\mathcal{F}_{\mathsf{CT}}^{\mathsf{stp}}$ and $\mathcal{F}_{\mathsf{CT}}^{\mathsf{sk}}$ returning respectively (Q, \mathbf{z}_0, g) and $(d_{\delta})_{\delta=0}^{\kappa-1}$ with $\mathbf{z}_0 \sim U(\mathbb{F}_q^2)$, $g \sim U(\mathbb{G}_1), d_{\delta} \sim U(\mathbb{G}_2)$ and $Q \subseteq [N], |Q| = \ell$ such that the probability of Q containing only corrupted users is smaller than $2^{-\lambda}$. Notice that t < N/2 implies $\ell \leq \lambda$.

In steps 1-6 members of the committee sample a polynomial f_i used for the VSS, and shares $\mathbf{w}_{i,\alpha}$. The proof in line 4 guarantees that the adversary is aware of the plaintext $f_i(j)$ encrypted, preventing decryption-oracle attacks.

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

Call $C, R, S, \mathcal{K} \leftarrow \emptyset, n \leftarrow 0$. Send setup to $\mathcal{F}_{\mathsf{SK}}$, wait for the reply (input, pp, f(i)) and parse pp = mpk, $g, h, k_0, \ldots, k_{N-1}$, with mpk = $[\mathbf{a}]_1$, $[\mathbf{a}^\top \mathbf{w}_1]_1$, $[\mathbf{a}^\top \mathbf{w}_2]_1$, $[\mathbf{a}^\top \mathbf{w}_3]_1$ together with bilinear groups description. Upon receiving:

- 1: (register): Send keygen to \mathcal{F}_{SK} and wait for (key, sk_n); Add $\mathcal{K} \leftarrow \mathcal{K} \cup sk_n$
- 2: $(\text{key}_{\text{requested}}, j)$ from \mathcal{F}_{SK} : Wait for all elected leader to reveal themselves.
- $3: \qquad \text{While } n \in S, \, n \leftarrow n+1; \quad R \leftarrow R \cup \{(j,n)\}, \, n \leftarrow n+1; \quad \text{Return (registered}, j)$
- 4: (elect, *eid*): Send (toss, *eid*) to \mathcal{F}_{CT}^{ch}
- 5: (tossed, eid, G, H, Q) from \mathcal{F}_{CT}^{ch} , if $i \in Q$:
- 6: Sample $s_i, r_i, \rho_i \leftarrow {}^{\$} \mathbb{F}_q$ and $m_i \leftarrow {}^{\$} [n]$ and compute:

7:
$$G_{1,i} \leftarrow g^{r_i} G^{m_i}, \quad G_{2,i} \leftarrow g^{\rho_i}, \quad \mathbf{c}_{0,i} \leftarrow [s_i \mathbf{a}]_1, \quad c_{1,i} \leftarrow h^{r_i} H^{m_i} \cdot [s_i \mathbf{a}^\top \mathbf{w}_1]_1$$

- 8: $c_{2,i} \leftarrow h^{-\rho_i} \cdot \left[s_i \mathbf{a}^\top \mathbf{w}_2 \right]_1, \quad c_{3,i} \leftarrow h^{-n\rho_i} \cdot \left[s_i \mathbf{a}^\top \mathbf{w}_3 \right]_1$
- 9: $\pi_{\mathsf{LR},i} \leftarrow \mathsf{NIZK}.\mathsf{P}_{\mathsf{LR}}(\mathbf{S}, (\mathbf{c}_{0,i}, c_{1,i}, c_{2,i}, c_{3,i}, G_{1,i}, G_{2,i}), [n], (s_i, r_i, \rho_i, m_i))$
- 10: Broadcast (msg, eid, $\mathbf{c}_{0,i}$, $c_{1,i}$, $c_{2,i}$, $c_{3,i}$, $G_{1,i}$, $G_{2,i}$, $\pi_{\mathsf{LR},i}$)
- 11: $(\mathsf{msg}, eid, \mathbf{c}_{0,\nu}, c_{1,\nu}, c_{2,\nu}, c_{3,\nu}, G_{1,\nu}, G_{2,\nu}, \pi_{\mathsf{LR},\nu})$ from P_{ν} :
- 12 : Let $Q_0 \subseteq Q$ be the set of ν such that $\pi_{\mathsf{LR},\nu}$ is accepted

13:
$$G_1 \leftarrow \prod_{\nu \in Q_0} G_{1,\nu}, \quad G_2 \leftarrow G \cdot \prod_{\nu \in Q_0} G_{2,\nu}, \quad K_{1,i} \leftarrow G_1^{f(i)}, \quad K_{2,i} \leftarrow G_2^{f(i)}$$

- 14: $\pi_{\mathsf{DDH},i} \leftarrow \mathsf{NIZK}.\mathsf{P}_{\mathsf{DDH}}((g,G_1,G_2),(k_i,K_{1,i},K_{2,i}),f(i))$
- 15: Broadcast (open, $eid, K_{1,i}, K_{2,i}, \pi_{\text{DDH},i}$)
- 16: (open, $eid, K_{1,\nu}, K_{2,\nu}, \pi_{\mathsf{DDH},\nu}$) for $\nu \in \mathbb{Z}, |\mathbb{Z}| = t$ with accepting $\pi_{\mathsf{DDH},\nu}$:
- 17: Reconstruct $K_j \leftarrow \prod_{\nu \in Z} K_{j,\nu}^{\lambda_{\nu}}$ with λ_{ν} the Lagrange coefficient for Z
- 18: $\mathbf{c}_0 \leftarrow \prod_{\mu \in Q_0} \mathbf{c}_{0,\mu}, \quad c_1 \leftarrow K_1^{-1} \cdot \prod_{\mu \in Q_0} c_{1,\mu}, \quad c_2 \leftarrow H^{-1}K_2 \cdot \prod_{\mu \in Q_0} c_{2,\mu}$
- 19: $c_3 \leftarrow H^{-n} K_2^n \cdot \prod_{\mu \in Q_0} c_{3,\mu}, \quad c \leftarrow (\mathbf{c}_0, c_1, c_2, c_3)$
- 20: If there exists $\mathsf{sk}_{\gamma,j} \in \mathcal{K}$ such that $1 \leftarrow \mathsf{FE}.\mathsf{Dec}(\mathbf{c}, \mathbf{y}_{\gamma,j}, \mathsf{mpk}, \mathsf{sk}_{\gamma,j})$:
- 21: Return (outcome, *eid*, 1) and store $C \leftarrow C \cup \{(eid, K_1, K_2)\}$
- 22 : **Else** return (outcome, eid, 0).
- 23: (reveal, *eid*): If there exists $(eid, K_1, K_2) \in C$: compute c as in steps 18, 19
- 24: Find $\mathsf{sk}_{\gamma,j} \in \mathcal{K}$ such that $1 \leftarrow \mathsf{FE}.\mathsf{Dec}(c, \mathbf{y}_{\gamma,j}, \mathsf{mpk}, \mathsf{sk}_{\gamma,j})$

25: Get
$$\pi \leftarrow \mathsf{NIZK}.\mathsf{P}_{\mathsf{Dec}}(\mathsf{mpk}, c, \mathbf{y}_{\gamma, j}, \mathsf{sk}_{\gamma, j})$$
 and send (claim, $eid, \pi, K_1, K_2, \gamma, j$)

- 26 : **Else** broadcast (claim, eid, \perp)
- 27: (claim, $eid, \pi, K_1, K_2, \gamma, \delta$) from P_{ν} : compute c as in steps 18, 19
- 28 : If $(\nu, \gamma) \in R$ and $1 \leftarrow \mathsf{NIZK}.\mathsf{V}_{\mathsf{Dec}}(\mathsf{mpk}, c, \mathbf{y}_{\gamma, \delta}, \pi)$: Send $(\mathsf{update}, \gamma, n)$ to $\mathcal{F}_{\mathsf{SK}}$
- 29: Update $S \leftarrow S \cup \{\gamma + \delta'n : \delta' \in [\kappa]\}$ and return (result, *eid*, ν)
- 30 : Else: return (rejected, eid, ν)

Fig. 4. Protocol $P_{\mathsf{SSLE},\kappa}^{(i)}$. $\mathbf{S} \in \mathbb{G}_1^{7,4}$ represents the linear operations in lines 6-8.

In lines 7-15 users test the VSS by checking if the exponents of h_{μ} , $(k_{i,\mu})_{i=0}^{N-1}$ lies in the right Reed-Solomon code. A standard test is to check orthogonality with a codeword in the dual space $\mathsf{RS}_{\mathbb{F},N+1,t}^{\perp}$. Next, consistency with $s_{i,\mu} = f_{\mu}(i)$ and $k_{i,\mu}$ is checked. If it fails the player will complain (lines 10-13) and remove P_{μ} from the committee.

Next, the generation of a new secret key begins by querying \mathcal{F}_{CT}^{sk} , line 20, which returns $(d_{\delta})_{\delta=0}^{k-1}$, interpreted as the randomness of requested OFE keys. In lines 21-25 members of the committee generate the secret key share $\mathbf{d}_{n,\delta}^{(i)}$ and privately send it to the receiver. Again a NIZK is added to prevent any decryption-oracle attack.

Observe now that, for every $\mu \in Q$

$$(\mathbf{z}_0, z_{1,\mu}, z_{2,\mu}, z_{2,\mu}) = [\mathbf{a}]_1, [\mathbf{a}^\top \mathbf{w}_{1,\mu}]_1, [\mathbf{a}^\top \mathbf{w}_{2,\mu}]_1, [\mathbf{a}^\top \mathbf{w}_{3,\mu}]_1$$

is a master public for our OFE scheme, and $(\mathbf{d}_{n,\delta}^{(\mu)}, d_{\delta})$ is a secret key for $(1, n, \delta)$ in the same scheme. Hence the recipient, lines 26-31, verifies this key share by checking if it is able to decrypt an encryption of **0**. Somewhat surprisingly in the proof of security we show that this is enough to ensure correctness of the key.

Finally, if the above check fails, the recipient broadcast a complaint message exposing the malformed key. Every user then checks the complaint and, if legitimate, remove P_{μ} from the committee.

Party $P_{SK}^{(i)}$ realising \mathcal{F}_{SK} :

	Initially set $n \leftarrow 0, S \leftarrow \emptyset$. Create $(pk_i, sk_i) \leftarrow^{\$} AE.Setup(1^{\lambda})$, broadcast $(user_key, pk_i)$, send $(toss) \rightarrow \mathcal{F}_{CT}^{stp}$ and wait for its reply $(tossed, Q, \mathbf{z}_0, g)$
1:	If $i \in Q$: Sample $f_i \leftarrow^{\$} \mathbb{F}_q[x]_{< t}, \mathbf{w}_{1,i}, \mathbf{w}_{2,i}, \mathbf{w}_{3,i} \leftarrow^{\$} \mathbb{F}_q^2$
2:	Compute $h_i \leftarrow g^{f_i(-1)}, k_{j,i} \leftarrow g^{f_i(j)}$ and $z_{\alpha,i} \leftarrow \mathbf{z}_0^{\mathbf{w}_{\alpha,i}}$ for $j \in [N], \alpha \in [3]$
3:	$c_{j,i} \leftarrow^{\$} AE.Enc(f_i(j),pk_j) \text{ with randomness } r_{j,i}$
4:	$\pi_{j,i} \leftarrow^{\$} NIZK.P_{Enc}(c_j,pk_j,f_i(j),r_{j,i})$
5:	Broadcast $(msg, h_i, (k_{j,i}, c_{j,i}, \pi_{j,i})_{j=0}^{N-1})$
6:	Send (prove, $(z_{\alpha,i})_{\alpha=1}^3, (\mathbf{w}_{\alpha,i})_{\alpha=1}^3)$ to $\mathcal{F}_{zk}^{\mathcal{R}_{Lin}}$
7:	When $P_{\mu} \to (msg, h_{\mu}, k_{j,\mu}, c_{j,\mu}, \pi_{j,\mu})_{j=0}^{N-1}$, $\mathcal{F}_{zk}^{\mathcal{R}_{Lin}} \to (proof, \mu, z_{\alpha,\mu})_{\alpha=1}^3$ for $\mu \in Q_0$:
8:	Set $\mathbf{k}_{\mu} = (h_{\mu}, k_{0,\mu}, \dots, k_{N-1,\mu})$ and sample $\mathbf{v} \leftarrow^{\$} RS_{\mathbb{F},N+1,t}^{\perp}$.
9:	If $\mathbf{k}^{\mathbf{v}}_{\mu} \neq 1$ or some $\pi_{j,\mu}$ is rejected: remove μ from Q_0 .
10:	Decrypt $s_{i,\mu} \leftarrow AE.Dec(c_{i,\mu},pk_i,sk_i)$. If $g^{s_{i,\mu}} \neq k_{i,\mu}$:
11:	$\pi \leftarrow NIZK.P_{Dec}(pk_i, c_{i,\mu}, s_{i,\mu}, sk_i), \text{ and broadcast } (complain, s_{i,\mu}, \mu, \pi)$
12:	Upon receiving (complain, μ , $s_{i,\mu}$, π) from P_j :
13:	If π is accepting and $g^{s_{i,\mu}} \neq k_{i,\mu}$, remove μ from Q_0 .
14:	Compute and store $z_{\alpha} \leftarrow \prod_{\mu \in Q_0} z_{\alpha,\mu}, h \leftarrow \prod_{\mu \in Q_0} h_{\mu}, k_j \leftarrow \prod_{\mu \in Q_0} k_{j,\mu}$
15:	$mpk \leftarrow (\mathbf{z}, z_1, z_2, z_3), pp \leftarrow (mpk, h, k_0, \dots, k_{N-1}), s_i \leftarrow \prod_{\mu \in Q_0} s_{i,\mu}$
	Fig. 5. Realisation \mathcal{F}_{SK} , Initial setup phase

Theorem 3 Protocol $\{P_{\mathsf{SK}}^{(i)} : i \in [N]\}$ securely realises $\mathcal{F}_{\mathsf{SK}}$ in the $(\mathcal{F}_{\mathsf{CT}}, \mathcal{F}_{\mathsf{zk}})$ hybrid model under the SXDH assumption for the class of PPT environments \mathcal{Z} that statically corrupt up to $\lfloor N/2 \rfloor$ players.

Party $P_{\mathsf{SK}}^{(i)}$ realising $\mathcal{F}_{\mathsf{SK}}$ upon receiving:

- 17: (setup): Return (input, pp, s_i)
- 18: (update, n, γ): set $S \leftarrow \{\gamma + \delta n : \delta \in [\kappa]\}$
- 19: (keygen): Broadcast (key_request)

20: (key_request) from P_j : Send (toss, rid|j) to \mathcal{F}_{CT}^{sk} and return (key_requested, j)

21: $(\mathsf{tossed}, rid|j, (d_{\delta})_{\delta=0}^{\kappa-1})$ from $\mathcal{F}_{\mathsf{CT}}^{\mathsf{sk}}$, if $i \in Q$:

- 22: While $n \in S$, increase $n \leftarrow n+1$
- $23: \quad \mathbf{d}_{n,\delta}^{(i)} \leftarrow \left[\mathbf{w}_{1,i} + n\mathbf{w}_{2,i} + \delta\mathbf{w}_{3,i}\right]_{d_{\delta}}, \quad \mathbf{d}_{n}^{(i)} \leftarrow (\mathbf{d}_{n,\delta}^{(i)})_{\delta=0}^{\kappa-1}$
- 24 : $c_i \leftarrow {}^{\$} \mathsf{AE}.\mathsf{Enc}(\mathbf{d}_n^{(i)},\mathsf{pk}_j)$ with randomness r_i
- 25: $\pi_i \leftarrow^{\$} \mathsf{NIZK}.\mathsf{P}_{\mathsf{Enc}}(c_i,\mathsf{pk}_j,\mathbf{d}_n^{(i)},r_i); \operatorname{Broadcast}(\mathsf{key_partial},c_i,\pi_i,j,n)$
- 26: (key_partial, c_{μ}, π_{μ}, i, n) with accepting π_{μ} from P_{μ} for $\mu \in Q_n$:
- 27: for all $\mu \in Q_n$ get $(\mathbf{d}_{n,\delta}^{(\mu)})_{\delta=0}^{\kappa-1} \leftarrow \mathsf{AE}.\mathsf{Dec}(c_i,\mathsf{sk}_i)$
- 28: If $e(\mathbf{z}_0, \mathbf{d}_{n,\delta}^{(\mu)}) \neq e(z_{1,\mu} \cdot z_{2,\mu}^n \cdot z_{3,\mu}^\delta, d_\delta)$:
- 29: Remove μ from $Q_n, \pi \leftarrow \mathsf{NIZK}.\mathsf{P}_{\mathsf{Dec}}(\mathsf{pk}_i, c_\mu, (\mathbf{d}_{n,\delta}^{(\mu)})_{\delta=0}^{\kappa-1}, \mathsf{sk}_i)$
- 30: Broadcast (key_complain, $\mu, n, \delta, (\mathbf{d}_{n,\delta}^{(\mu)})_{\delta=0}^{\kappa-1}, \pi$)
- 31: Set $\mathsf{sk}_{n,\delta} \leftarrow \left(\prod_{\mu \in Q_n} \mathbf{d}_{n,\delta}^{(\mu)}, d_{\delta}\right)$ and return $(\mathsf{key}, (\mathsf{sk}_{n,\delta})_{\delta=0}^{\kappa-1})$
- 32: (key_complain, μ, n, δ , $(\mathsf{sk}_{n,\delta}^{(\mu)})_{\delta=0}^{\kappa-1}, \pi$) from P_j with accepting π :
- 33: Perform the test on line 28. If it fails:
- 34: Remove μ from Q_n and for each key received sk let \mathbf{d}_{μ} be P_{μ} 's share
- 35: Parse sk = (d, d), return (key_update, (d \cdot d_µ⁻¹, d))

Fig. 6. Realisation $\mathcal{F}_{\mathsf{SK}},$ Key Distribution phase

6 Efficiency considerations

Overall communication costs of our protocol are summarised in Table 2. As mentioned in the previous section, however most of these messages are not required for verification and, in particular, they do not need to be stored on chain.

More in detail, for the VSS to generate the ElGamal public and secret keys, only aggregated elements h, k_0, \ldots, k_{N-1} have to be placed on-chain, as those are the only ones required to verify the secret sharing. Next, during elections, we have to store the partial ciphertexts and related NIZKs sent by the committee, as these components are necessary to reconstruct the election's ciphertext. However, our specific OFE and protocol allow the winner to aggregate the expensive threshold decryption, without the need to also post a proof of correctness. Note that the same property does not hold for the first round, since together with the partial ciphertexts one would have to aggregate the corresponding NIZKs with more sophisticated tools. Finally we remark that it is also possible to avoid storing encrypted secret keys for our OFE on chain, using the chain only for disputes.

As shown in the Table, while election requires low communications, the setup is more expensive, requiring 252 MB for 2^{14} users. However, this is supposed to be performed rarely. Once this is done, our protocol allows new users to join providing them new secret key, without updating the key material of other users. This registration takes only 73 KB of communication. Letting users leave

the system on the other hand introduces some inefficiencies. The problem is that users who go away may still be elected, causing some elections to end without a winner. An obvious, but expensive, way to completely remove this problem is to perform a new setup every time that one or more users leave. However, one can also make a trade-off leaving the possibility that some elections finish without a winner, and redo the setup only when this probability (which for L inactive users out of N registered users is L/N) becomes too high.

Procedure	Number of elements sent			Size		
-	\mathbb{F}_q	\mathbb{G}_1	\mathbb{G}_2	\mathbb{G}_T	off- $chain$	on- $chain$
VSS for ElGamal	$2\lambda N$	$2\lambda N + \lambda$	_	_	$252 \mathrm{MB}$	1.05 MB
${\rm Distribute}~{\sf mpk}$	2λ	5λ	_	_	$30.7~\mathrm{KB}$	$30.7~\mathrm{KB}$
Election, 1 st Round	$\kappa(6 + 2\log n)$	$\kappa(10 + \log n)$	_	_	36.8 KB	36.8 KB
Election, 2 nd Round	t+1	5(t+1)	-	_	2.88 MB	-
Election, Claim	2	2	1	2	$992 \mathrm{B}$	$992 \mathrm{B}$
Registration	λ	_	$2\kappa\lambda+2\lambda$	_	73.3 KB	_

Table 2. Communication costs of our scheme, using ElGamal in place of the generic IND-CPA encryption. Size is computed assuming $\log |\mathbb{F}_q| = 256$, $\log |\mathbb{G}_1| = 512$, $\log |\mathbb{G}_2| = 256$, $\log |\mathbb{G}_T| = 3072$, $\lambda = 80$, $\kappa = \log N$, $t = \lfloor N/2 \rfloor$ and $N = 2^{14}$.

Comparison with [BEHG20] We now compare our UC-secure construction with the shufflebased solution in [BEHG20], which we briefly recall here. Essentially the public state contains a list of Diffie-Hellman pairs $(K_{i,1}, K_{i,2})$, one for every user, and P_i 's secret key is a discrete log k_i such that $K_{i,2} = K_{i,1}^{k_i}$. An election is performed by choosing one of those tuples through the random beacon and the leader claims victory by revealing its secret key. To achieve unpredictability, each time a pair is added by a user, he sends a shuffled and re-randomized list along with a NIZK. Note that every election involves at least the registration of the previous winner, who has "burnt" her secret key, if she desires to stay. Moreover, this implies that the protocol requires at each round as many shuffles as the number of new users. Notably, all the lists and NIZKs have to be posted on chain in order to ensure verifiability.

In the high communication solution, denoted N-shuffle, each shuffles costs 2n group elements, while the more efficient and less secure one, denoted \sqrt{N} -shuffle, costs $2\sqrt{n}$ elements.

In light of the requirement in [Lab19] to support $O(\log^2 N)$ new users per round, we compare these solutions evaluating the cumulative cost of several elections, interleaving between every two a fixed amount of registrations. In Fig. 6 we provide the communication costs for such a scenario where we assume to start with 2^{14} users and then perform: 10 registrations for each election in the first column, 20 in the second column, and 50 in the third one.

We remark that in those plots, the costs of the shuffle-based solutions do not even include the costs of setup¹¹, as it can be done only once in contrast to ours where we need to occasionally refresh the secret key material. In spite of that, the cost of our setup is quickly compensated by our lighter registration and election, which makes our solution more suited to dynamic scenarios.

¹¹ I.e. the cost to generate a shuffled list containing the pairs of the initial users. This has cost $O(n^2)$ if everyone performs a shuffle, or $O(\kappa n)$ using an approach similar to ours where a random committee of κ users shuffle the initial list

More efficient SSLE with Game Based Security We now remark that communication complexity can be further reduced in our construction at the cost of giving up UC security yet achieving the game-based security notion.

As we would not need any more to simulate each election, every secret key can now be produced without artificially skipping some of them. For the same reason, the NIZKs need not to be simulation-extractable, which allow us to use Bulletproofs for the range proofs. This reduces on-chain costs to $O(\kappa \log \log N)$.

Finally, when giving up UC security users who voluntarily leave the system can be handled by asking such users to reveal their own secret keys upon leaving, as done in [BEHG20]. This way, if a revoked user happens to be elected, everyone can detect it and immediately proceed to generate a new election's ciphertext. To keep round complexity low, one can also prepare several challenges per election, order them, remove those that can be decrypted with keys of users who left, and set the current challenge as the first of the remaining ones. This solution only works for non-UC security though, as the simulator should now generate on request honest user's secret key that are consistent with previous elections.

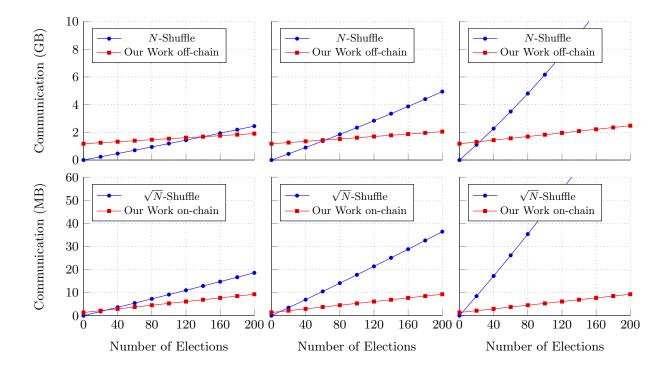


Fig. 7. Cumulative communication costs in this work and [BEHG20]. Initially the number of users is $N = 2^{14}$ and between every two elections 10 (left column), 20 (middle column) or 50 (right column) registrations occur.

Acknowledgements

This work has received funding in part from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program under project PICOCRYPT (grant

agreement No. 101001283), by the Spanish Government under projects SCUM (ref. RTI2018-102043-B-I00), CRYPTOEPIC (ref. EUR2019-103816), and SECURITAS (ref. RED2018-102321-T), by the Madrid Regional Government under project BLOQUES (ref. S2018/TCS-4339), by a research grant from Nomadic Labs and the Tezos Foundation and by the Programma ricerca di ateneo UNICT 2020-22 linea 2.

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 Extensions. In V. Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 205–222. Springer, Heidelberg, August 2005.
- 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.
- 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.
- 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.
- 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.
- 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.
- 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.
- 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.
- Lab19. P. Labs. Secret single-leader election (SSLE). https://web.archive.org/web/20191228170149/https: //github.com/protocol/research-RFPs/blob/master/RFPs/rfp-6-SSLE.md, 2019.
- 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

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

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 $a \in A$, $\Pr[x_{|A} = a] = \Pr[x = a|A]$. 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

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

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 $A \subseteq S$

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

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

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

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}_{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 = 5, 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 8 satisfies perfect completeness, special soundness and perfect HVZK

$\mathcal{P}_{Dec}(mpk, c, \mathbf{y}, sk):$		$\mathcal{V}_{Dec}(mpk, c, \mathbf{y})$:	
Parse $mpk = (\mathbf{k}, k^*, k_i)_{i=1}^n$,	Parse $mpk = (\mathbf{k}, k^*, k_i)_{i=1}^n$,		
$c = (\mathbf{c}, c_i)_{i=1}^n, \mathbf{y} = (y_i)_{i=1}^n,$		$c = (\mathbf{c}, c_i)_{i=1}^n, \mathbf{y} = (y_i)_{i=1}^n,$	
$sk = (\mathbf{d}, d)$			
$\mathbf{v} \leftarrow^{\$} \mathbb{G}_2^2, \rho \leftarrow^{\$} \mathbb{F}_q$			
$\widehat{\mathbf{d}} \leftarrow \mathbf{d}^{\rho}, \widehat{d} \leftarrow d^{\rho}$			
$T_0 \leftarrow e(\mathbf{k}, \mathbf{v})$			
$T_1 \leftarrow e(\mathbf{c}, \mathbf{v})$	$\xrightarrow{\widehat{d}, T_0, T_1}$		
	\leftarrow r	$r \leftarrow^{\$} \mathbb{F}_q$	
$\mathbf{z} \leftarrow \widehat{\mathbf{d}}^r \cdot \mathbf{v}$	$\xrightarrow{\mathbf{Z}}$		
		Check:	
		$\widehat{d} \neq 1_{\mathbb{G}_2}$	
		$e(\mathbf{k}, \mathbf{z}) \stackrel{?}{=} T_0 \cdot e(k_1^{y_1}, \dots, k_n^{y_n}, \widehat{d})^r$	
		$e(\mathbf{c}, \mathbf{z}) \stackrel{?}{=} T_1 \cdot e(c_1^{y_1}, \dots, c_n^{y_n}, \widehat{d})^r$	
		$e(\mathbf{c},\mathbf{z}) = 1_1 \cdot e(c_1,\ldots,c_n,d)$	

Fig. 8. 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 $\mathsf{mpk} = (\mathbf{k}, k^*, k_1, \ldots, k_n)$ and $\mathsf{sk} = (\mathbf{d}, d)$ with

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

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

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

Special Soundness: Given $(\hat{d}, T_0, T_1, r_1, \mathbf{z}_1)$ and $(\hat{d}, T_0, T_1, r_2, \mathbf{z}_2)$ two accepting transcripts, calling $\hat{\mathbf{d}} = (\mathbf{z}_1 \cdot \mathbf{z}_2^{-1})^{(r_1 - r_2)^{-1}}$ 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 $\mathsf{FE}.\mathsf{Dec}(c, \mathbf{y}, \mathsf{sk}, \mathsf{mpk}) = 1$. We show now that the first equation implies $(\mathsf{mpk}, \mathbf{y}, \mathsf{sk}) \in \mathcal{L}_{\mathsf{key}}$. Fixed $\mathsf{mpk} = (\mathbf{k}, k^*, k_1, \ldots, k_n)$ with

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

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 $(\bar{\mathbf{c}}, \bar{c}_1, \dots, \bar{c}_n)$ be the encryption of a vector \mathbf{m} with randomness $s \neq 0$. By construction

$$\begin{split} \bar{\mathbf{c}} &= \left[s\mathbf{a}\right]_{1}, \quad \bar{c}_{i} = \left[s\mathbf{a}(m_{i}\mathbf{u} + \mathbf{w}_{i})\right]_{1} \quad \Rightarrow \quad e(\bar{\mathbf{c}}, \widehat{\mathbf{d}}) \cdot e(\bar{c}_{1}^{y_{1}} \cdot \ldots \cdot \bar{c}_{n}^{y_{n}}, \widehat{d})^{-1} \\ &= \left[s\mathbf{a}^{\top} + s\mathbf{a}^{\top} \sum_{i=1}^{n} \sigma y_{i} m_{i}\mathbf{u} - s\mathbf{a}^{\top} \sum_{i=1}^{n} \sigma y_{i} \mathbf{w}_{i}\right]_{T} = \left[s(\mathbf{a}^{\top}\mathbf{u})(\mathbf{y}^{\top}\mathbf{m})\right]_{T}. \end{split}$$

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 S_{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 $S_{Dec}(mpk, c, y)$:

- 1: Parse mpk = $(\mathbf{k}, k^*, k_1, \dots, k_n), \quad c = (\mathbf{c}, c_1, \dots, c_n), \quad \mathbf{y} = (y_1, \dots, y_n)$
- 2: Sample $r' \leftarrow^{\$} \mathbb{F}_q$, $\widehat{d'} \leftarrow^{\$} \mathbb{G}_2$, $\mathbf{z}' \leftarrow^{\$} \mathbb{G}_2^2$
- 3: $T'_0 \leftarrow e(\mathbf{k}, \mathbf{z}') \cdot e(k_1^{y_1} \cdot \ldots \cdot k_n^{y_n}, \widehat{d}')^{-r}, \quad T'_1 \leftarrow e(\mathbf{c}, \mathbf{z}') \cdot e(c_1^{y_1} \cdot \ldots \cdot c_n^{y_n}, \widehat{d}')^{-r}$
- 4: Return $(\hat{d}', T'_0, T'_1, r', \mathbf{z}')$

 $(\hat{d}', T'_0, T'_1, r', \mathbf{z}')$ 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 $(\mathbf{z}, \hat{d}, r) \sim U(\mathbb{G}_2^3 \times \mathbb{F}_q)$ and $(\mathbf{z}, \hat{d}', r') \sim U(\mathbb{G}_2^3 \times \mathbb{F}_q)$. 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 $\forall \mathbf{z}_0, \hat{d}_0, r_0$, upon conditioning on $\hat{d} = \hat{d}_0 = \hat{d}', \mathbf{z} = \mathbf{z}_0 = \mathbf{z}'$ 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

$$T_{0} = e(\mathbf{k}, \mathbf{v}) = e(\mathbf{k}, \mathbf{z}_{0}) \cdot e(\mathbf{k}, \mathbf{d}^{\rho})^{-r_{0}} =$$

= $e(\mathbf{k}, \mathbf{z}_{0}) \cdot e(k_{1}^{y_{1}} \cdot \ldots \cdot k_{n}^{y_{n}}, d^{\rho})^{-r_{0}} =$
= $e(\mathbf{k}, \mathbf{z}_{0}) \cdot e(k_{1}^{y_{1}} \cdot \ldots \cdot k_{n}^{y_{n}}, \widehat{d}_{0})^{-r_{0}} = T_{0}'$

Analogously $T_1 = T'_1$ and therefore $\Delta((T_0, T_1), (T'_0, T'_1)) = 0$.

B.2 Linear relation in the exponent

We now provide a sigma protocol for \mathcal{R}_{LR} that proves, given $[A]_1$, $[\mathbf{b}]_1$ and an integer ℓ , the knowledge of an \mathbf{x} such that $A\mathbf{x} = \mathbf{b}$ with the last coordinate $x_n \in \{0, \ldots, 2^{\ell} - 1\}$. To this aim we first provide a randomised sigma protocol for the simpler relation

$$\mathcal{R}_{\mathsf{Lin}} = \{(([A]_1, [\mathbf{b}]_1), \mathbf{x}) \ : \ \mathbf{A} \in \mathbb{F}_q^{m, n}, \ \mathbf{x} \in \mathbb{F}_q^n, \ \mathbf{b} \in \mathbb{F}_q^m, \ \mathbf{A}\mathbf{x} = \mathbf{b}\}$$

A first solution can be derived from [Mau15] where the prover sends m group elements and n field elements. In the following we provide a more efficient protocol requiring only 3 group elements and m + 1 field elements.

$\mathcal{P}_{Lin}([A]_1,[\mathbf{b}]_1,\mathbf{x}):$		$\mathcal{V}_{Lin}(\left[A ight]_{1},\left[\mathbf{b} ight]_{1}):$
	$\underbrace{u_0,\ldots,u_n}$	$u_0,\ldots,u_n \leftarrow^{\$} \mathbb{G}_1$
$\alpha \leftarrow^{\$} \mathbb{F}_{q}, \ c \leftarrow u_{0}^{\alpha} \cdot \prod_{i=1}^{n} u_{i}^{x_{i}}$	\xrightarrow{c}	
	r ←	$\mathbf{r} \leftarrow^{\$} \mathbb{F}_q^n$
$\mathbf{y} \leftarrow^{\$} \mathbb{F}_{q}^{n}, \ \beta \leftarrow^{\$} \mathbb{F}_{q}$ $v_{1} \leftarrow [\mathbf{r}^{\top} \mathbf{A} \mathbf{y}]_{1}$ $v_{2} \leftarrow u_{0}^{\beta} \cdot \prod_{i=1}^{n} u_{i}^{y_{i}}$		
	$\xrightarrow{(v_1, v_2)}$	
	$\stackrel{\rho}{\longleftarrow}$	$\rho \leftarrow^{\$} \mathbb{F}_q$
$\mathbf{w} \leftarrow (\mathbf{y} + \rho \mathbf{x}, \beta + \rho \alpha)$	$\overset{\mathbf{W}}{-\!\!\!-\!\!\!-\!\!\!-\!\!\!-\!\!\!-\!\!\!-\!\!\!-\!\!\!-\!\!\!-$	
		Parse $\mathbf{w} = \mathbf{z} \gamma$ and check:
		$\begin{bmatrix} \mathbf{r}^{\top} A \mathbf{z} \end{bmatrix}_1 \stackrel{?}{=} v_1 \cdot \begin{bmatrix} \mathbf{r}^{\top} \mathbf{b} \end{bmatrix}_1^{ ho}$
		$u_0^{\gamma} \cdot \prod_{i=1}^n u_i^{z_i} \stackrel{?}{=} v_2 \cdot c^{\rho}$

Fig. 9. Randomised sigma protocol for the \mathcal{R}_{Lin} relation

Proposition 3. If the Discrete Logarithm Problem is hard in \mathbb{G}_1 , the protocol described in Figure 9 is a constant round, public coin argument of knowledge with perfect completeness and HVZK.

Proof. Completeness. If **x** is such that $A\mathbf{x} = \mathbf{b}$ then for any $\mathbf{r} \in \mathbb{F}_q^m$ and ρ we have that

$$\begin{bmatrix} \mathbf{r}^{\top} A \mathbf{z} \end{bmatrix}_{1} = \begin{bmatrix} \mathbf{r}^{\top} A (\mathbf{y} + \rho \mathbf{x}) \end{bmatrix}_{1} = \begin{bmatrix} \mathbf{r}^{\top} A \mathbf{y} \end{bmatrix}_{1} \cdot \begin{bmatrix} \mathbf{r}^{\top} A \mathbf{x} \end{bmatrix}_{1}^{\rho} = v_{1} \cdot \begin{bmatrix} \mathbf{r}^{\top} \mathbf{b} \end{bmatrix}_{1}^{\rho}$$
$$u_{0}^{\gamma} \cdot \prod_{i=1}^{n} u_{i}^{z_{i}} = u_{0}^{\beta} \cdot \prod_{i=1}^{n} u_{i}^{y_{i}} \cdot u_{0}^{\rho\alpha} \cdot \prod_{i=1}^{n} u_{i}^{\rho x_{i}} = v_{2} \cdot c^{\rho}.$$

HVZK: We sketch a simulator \mathcal{S}_{Lin} that produce a correctly distributed transcript. On input $[A]_1$, $[\mathbf{b}]_1$, it samples $\mathbf{u} \leftarrow^{\$} \mathbb{G}_1^{n+1}$, $c \leftarrow^{\$} \mathbb{G}_1$, $\mathbf{r} \leftarrow^{\$} \mathbb{F}_q^m$, $\rho \leftarrow^{\$} \mathbb{F}_q$ and $\mathbf{w} \leftarrow^{\$} \mathbb{F}_q^{n+1}$. Then it parse $\mathbf{w} = \mathbf{z} | \gamma$ and compute

$$v_1 \leftarrow \left[\mathbf{r}^\top A \mathbf{z}\right]_1 \cdot \left[\mathbf{r}^\top \mathbf{b}\right]_1^{-\rho}, \quad v_2 \leftarrow \mathbf{u}^{\mathbf{w}} \cdot c^{-\rho}.$$

To see that the output follows the same distribution of an honestly generated transcript we first condition on the verifier's message, uniformly random in both cases. Next we observe that, as $\alpha \sim U(\mathbb{F}_q)$, in the honest view *c* is uniform. Conditioning on *c*, since **y**, **b** are distributed uniformly and independently from the previous variables, we have that $\mathbf{w} \sim U(\mathbb{F}_q^{n+1})$. Finally in both cases conditioning on **w** we have that v_1, v_2 are uniquely determined as the elements that makes the transcript accepting. **Proof of knowledge**. Given a malicious prover $\widetilde{\mathcal{P}}$ using at most θ random bits, we design an extractor \mathcal{E} that initially samples a random tape $R \leftarrow^{\$} \{0,1\}^{\theta}$ and $\mathbf{u} \leftarrow^{\$} \mathbb{G}_{1}^{n+1}$ and set $c \leftarrow \widetilde{\mathcal{P}}(\mathbf{u}; R)$. Then samples $\mathbf{r}_{i} \leftarrow^{\$} \mathbb{F}_{q}^{m}$ and $\rho_{i,j}$ for $i, j \in \{0,1\}$ and runs the malicious prover to get

$$(v_{i,1}, v_{i,2}) \leftarrow \widetilde{\mathcal{P}}(\mathbf{r}_i, \mathbf{u}; R), \qquad \mathbf{z}_{i,j} | \gamma_{i,j} \leftarrow \widetilde{\mathcal{P}}(\rho_{i,j}, \mathbf{r}_i; R)$$

and if the four transcript obtained are accepting, checks if

$$\mathbf{x}^* \coloneqq \frac{\mathbf{z}_{0,0} - \mathbf{z}_{0,1}}{
ho_{0,0} -
ho_{0,1}} \stackrel{?}{=} \frac{\mathbf{z}_{1,0} - \mathbf{z}_{1,1}}{
ho_{1,0} -
ho_{1,1}}.$$

If any of the transcripts is rejecting or if the last check fails, \mathcal{E} aborts, otherwise return \mathbf{x}^* .

In order to conclude we show that, calling η the probability that \mathcal{P} produce, if this quantity is non-negligible then \mathcal{E} returns a witness with non-negligible probability. Let $\mathsf{AT}(R, \mathbf{u}, \mathbf{r}, \rho)$ be the event that $\widetilde{\mathcal{P}}$ with randomness R, receiving \mathbf{u}, \mathbf{r} and ρ at each round, the transcript he produced is accepting. We further define

$$X = \{ (R, \mathbf{u}, \mathbf{r}_i, \rho_{i,j}) : \mathsf{AT}(R, \mathbf{u}, \mathbf{r}_i, \rho_{i,j}) \quad \forall i, j \in \{0, 1\} \}$$

Claim 1. Given a uniformly sampled vector, $\Pr[(R, \mathbf{u}, \mathbf{r}_i, \rho_{i,j}) \in X] \ge \eta^4$.

Proof of Claim: The above probability can be rewritten as $\Pr[\mathsf{AT}(R, \mathbf{u}, \mathbf{r}_i, \rho_{i,j}) \ i, j \in [2]] =$

$$= \sum_{\widetilde{R},\widetilde{\mathbf{u}}} \Pr\left[\mathsf{AT}(\widetilde{R},\widetilde{\mathbf{u}},\mathbf{r}_{i},\rho_{i,j}) \ i,j\in[2]\right] \Pr\left[R,\mathbf{u}=\widetilde{R},\widetilde{\mathbf{u}}\right]$$
$$= \sum_{\widetilde{R},\widetilde{\mathbf{u}}} \Pr\left[\mathsf{AT}(\widetilde{R},\widetilde{\mathbf{u}},\mathbf{r}_{0},\rho_{0,j}) \ j\in[2]\right] \Pr\left[\mathsf{AT}(\widetilde{R},\widetilde{\mathbf{u}},\mathbf{r}_{1},\rho_{1,j}) \ j\in[2]\right] \Pr\left[R,\mathbf{u}=\widetilde{R},\widetilde{\mathbf{u}}\right]$$
$$= \sum_{\widetilde{R},\widetilde{\mathbf{u}}} \Pr\left[\mathsf{AT}(\widetilde{R},\widetilde{\mathbf{u}},\mathbf{r},\rho_{j}) \ j\in[2]\right]^{2} \Pr\left[R,\mathbf{u}=\widetilde{R},\widetilde{\mathbf{u}}\right]$$
$$\geq \left(\sum_{\widetilde{R},\widetilde{\mathbf{u}}} \Pr\left[\mathsf{AT}(\widetilde{R},\widetilde{\mathbf{u}},\mathbf{r},\rho_{j}) \ j\in[2]\right] \Pr\left[R,\mathbf{u}=\widetilde{R},\widetilde{\mathbf{u}}\right]\right)^{2}$$

where the last step follows from the AM-QM inequality. In order to proceed we study the inner term $\Pr\left[\mathsf{AT}(\widetilde{R}, \widetilde{\mathbf{u}}, \mathbf{r}, \rho_j) \ j \in [2]\right] =$

$$= \sum_{\widetilde{\mathbf{r}}} \Pr\left[\mathsf{AT}(\widetilde{R}, \widetilde{\mathbf{u}}, \widetilde{\mathbf{r}}, \rho_j) \ j \in [2]\right] \Pr\left[\mathbf{r} = \widetilde{\mathbf{r}}\right]$$
$$= \sum_{\widetilde{\mathbf{r}}} \Pr\left[\mathsf{AT}(\widetilde{R}, \widetilde{\mathbf{u}}, \widetilde{\mathbf{r}}, \rho)\right]^2 \Pr\left[\mathbf{r} = \widetilde{\mathbf{r}}\right]$$
$$= \left(\sum_{\widetilde{\mathbf{r}}} \Pr\left[\mathsf{AT}(\widetilde{R}, \widetilde{\mathbf{u}}, \widetilde{\mathbf{r}}, \rho)\right] \Pr\left[\mathbf{r} = \widetilde{\mathbf{r}}\right]\right)^2$$
$$\geq \Pr\left[\mathsf{AT}(\widetilde{R}, \widetilde{\mathbf{u}}, \mathbf{r}, \rho)\right]^2$$

where the last inequality is again by AM-QM. Applying this to our first equation we get that

$$\dots = \left(\sum_{\widetilde{R},\widetilde{\mathbf{u}}} \Pr\left[\mathsf{AT}(\widetilde{R},\widetilde{\mathbf{u}},\mathbf{r},\rho)\right]^2 \Pr\left[R,\mathbf{u}=\widetilde{R},\widetilde{\mathbf{u}}\right]\right)^2$$
$$\geq \left(\sum_{\widetilde{R},\widetilde{\mathbf{u}}} \Pr\left[\mathsf{AT}(\widetilde{R},\widetilde{\mathbf{u}},\mathbf{r},\rho)\right] \Pr\left[R,\mathbf{u}=\widetilde{R},\widetilde{\mathbf{u}}\right]\right)^4$$
$$= \Pr\left[\mathsf{AT}(R,\mathbf{u},\mathbf{r},\rho)\right]^4 = \eta^4.$$

This concludes the claim's proof.

Next we define two other sets, namely Y as the set of $(R, \mathbf{u}, \mathbf{r}_i, \rho_{i,j})$ with $\mathbf{r}_0 \neq \mathbf{r}_1, \rho_{i,0} \neq \rho_{i,1}$ and Z the set of tuples in $X \cap Y$ such that calling c, $(v_{i,1}, v_{i,2})$ and $\mathbf{z}_{i,j}|_{\gamma_{i,j}}$ as in the construction of \mathcal{E} then

$$\frac{\mathbf{z}_{0,0} - \mathbf{z}_{0,1}}{\rho_{0,0} - \rho_{0,1}} \neq \frac{\mathbf{z}_{1,0} - \mathbf{z}_{1,1}}{\rho_{1,0} - \rho_{1,1}}$$

We informally observe that a random tuple lies in Z with negligible probability ε as the two different vectors above represent two distinct openings of the same Pedersen commitment c, that is computationally binding under the DL assumption.

Moreover the probability that a random vector does not lie in Y is, through a union bound, smaller that $|\mathbb{F}_q|^{-m} + |\mathbb{F}_q|^{-1} + |\mathbb{F}_q|^{-1} \leq 3q^{-1}$. We can therefore conclude that a random tuple lies in $T = X \cap Y \setminus Z$ with probability at least $\eta^4 - \varepsilon - 4q^{-1}$. If η is non-negligible, then so is this lower bound and in particular it is larger than $|\mathbb{F}|^{-1}$. As a consequence we will show that there are enough vectors **r** that make the extractor \mathcal{E} obtain the same witness. More formally we define the set

$$E = \{ (R, \mathbf{u}, \mathbf{r}, \rho_1, \rho_2) : \exists L < \mathbb{F}_q^m : (R, \mathbf{u}, \mathbf{r}, \mathbf{s}, \rho_1, \rho_2, \rho_1', \rho_2') \in T \Rightarrow \mathbf{s} \in L \}$$

Claim 2. $\Pr\left[(R, \mathbf{u}, \mathbf{r}_i, \rho_{i,j}) \in T, (\mathbf{r}_0, \rho_{0,j})_{j=0}^1 \notin E\right] \ge \eta^4 - \varepsilon - 4q^{-1}.$

Proof of Claim: Observe that the probability that a random tuple belongs to T conditioning to the event $(R, \mathbf{u}, \mathbf{r}_0, \rho_{0,j})_{j=0}^1 \in E$ is smaller that q^{-1} as, by the way we constructed E, this implies $\mathbf{r}_1 \in L$ that happens with probability $|L|q^{-m} \leq q^{-1}$. Therefore, calling for simplicity π the projection that sends $(R, \mathbf{u}, \mathbf{r}_i, \rho_{i,j})$ to $(R, \mathbf{u}, \mathbf{r}_0, \rho_{0,j})$, we have that

$$\eta^{4} - \varepsilon - 3q^{-1} \leq \Pr\left[\mathbf{v} \in T | \pi(\mathbf{v}) \in E\right] \Pr\left[\pi(\mathbf{v}) \in E\right] + \Pr\left[\mathbf{v} \in T, \ \pi(\mathbf{v}) \notin E\right]$$
$$\leq q^{-1} + \Pr\left[\mathbf{v} \in T, \ \pi(\mathbf{v}) \notin E\right].$$

The claim is therefore proven.

Finally, with probability $\eta^4 - \varepsilon - 4q^{-1}$, \mathcal{E} samples a tuple $(R, \mathbf{u}, \mathbf{r}_i, \rho_{i,j})$ in T with $(\mathbf{r}_0, \rho_{0,0}, \rho_{0,1}) \notin E$. By construction of E this means that there exists a base $\mathbf{s}_1, \ldots, \mathbf{s}_m \in \mathbb{F}_q^m$ and elements $\sigma_{h,j}$ for $h \in \{1, \ldots, m\}, j \in \{0, 1\}$ such that $(R, \mathbf{u}, \mathbf{r}_0, \mathbf{s}_h, \rho_{0,j}, \sigma_{h,j})_{j=0}^1 \in T$. Since $T \subseteq X$ all this random elements makes $\widetilde{\mathcal{P}}$ produce accepting transcripts.

More in detail, calling $(v_{h,1}, v_{h,2}) \leftarrow \widetilde{\mathcal{P}}(\mathbf{u}, \mathbf{s}_h; R)$, $\mathbf{z}_{h,j} | \gamma_{h,j} \leftarrow \widetilde{\mathcal{P}}(\mathbf{u}, \mathbf{s}_h, \rho_{h,j}; R)$ and $\mathbf{z}_{0,j} | \gamma_{0,j} \leftarrow \widetilde{\mathcal{P}}(\mathbf{u}, \mathbf{r}_0, \rho_{0,j})$,

$$\mathbf{x}^* = rac{\mathbf{z}_{0,0} - \mathbf{z}_{0,1}}{
ho_{0,0} -
ho_{0,1}} = rac{\mathbf{z}_{h,0} - \mathbf{z}_{h,1}}{\sigma_{h,0} - \sigma_{h,1}}$$

Moreover, since the transcripts are accepting

$$\begin{bmatrix} \mathbf{s}_{h}^{\top} A \mathbf{z}_{h,j} \end{bmatrix}_{1} = v_{h,1} \cdot \begin{bmatrix} \mathbf{s}_{h}^{\top} \mathbf{b} \end{bmatrix}_{1}^{\sigma_{h,j}} \Rightarrow \mathbf{s}_{h}^{\top} A \left(\frac{\mathbf{z}_{h,0} - \mathbf{z}_{h,1}}{\sigma_{h,0} - \sigma_{h,1}} \right) = \mathbf{s}_{h}^{\top} \mathbf{b} \Rightarrow \Rightarrow \mathbf{s}_{h}^{\top} (A \mathbf{x}^{*} - \mathbf{b}) = \mathbf{0} \Rightarrow A \mathbf{x}^{*} = \mathbf{b}$$

where the last equality follows as $\mathbf{s}_1, \ldots, \mathbf{s}_m$ is a base of \mathbb{F}_q^m by construction.

Next, we provide a constant round public coin proof for $\mathcal{R}_{\mathsf{LR}}$ that builds upon $\mathcal{R}_{\mathsf{Lin}}$. In addition to prove knowledge of a solution to the linear system we also have to prove that $x_n \in [2^{\ell}]$ for some given integer ℓ . The idea is to prove that x_n can be written as $\beta_0 + 2\beta_1 + \ldots + \beta_{\ell-1}2^{\ell-1}$ with $\beta_i \in \{0, 1\}$. This can be done by simply adding ℓ more variables and one equation to linear system, which leaves us only with the problem of proving $\beta_i \in \{0, 1\}$.

A standard technique is, given k, h_i random group element, to produce $c_i = k^{\beta_i} h_i^r$ a Pedersen commitment of β_i and then proving knowledge of an opening for β_i and β_i^2 . This is achieved by proving knowledge of β_i, r, s_i such that

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

where $s_i = r - b_i r$. In conclusion we produce overall a system with $2\ell + 1$ more equations and $2\ell + 1$ extra variables, and communicate ℓ commitments. For completeness we detail the protocol below.

 $\begin{array}{ccc} \underline{\mathcal{P}_{\mathsf{LR}}([A]_{1}, [\mathbf{b}]_{1}, 2^{\ell}, \mathbf{x}) :} & & \mathcal{V}_{\mathsf{LR}}([A]_{1}, [\mathbf{b}]_{1}, 2^{\ell}) : \\ & \underset{k, h_{0}, \dots, h_{\ell-1}}{\overset{k, h_{0}, \dots, h_{\ell-1}}{\leftarrow}} & k, h_{0}, \dots, h_{\ell-1} \leftarrow^{\$} \mathbb{G}_{1} \\ & \underset{k, h_{0}, \dots, h_{\ell-1}}{\overset{k, h_{0}, \dots, h_{\ell-1}}{\leftarrow}} & \underset{k, h_{0}, \dots, h_{\ell-1} \leftarrow^{\$} \mathbb{G}_{1} \\ & \underset{k, h_{0}, \dots, h_{\ell-1}}{\overset{k, h_{0}, \dots, h_{\ell-1}}{\leftarrow}} & \underset{k, h_{0}, \dots, h_{\ell-1} \leftarrow^{\$} \mathbb{G}_{1} \\ & \underset{k, h_{0}, \dots, h_{\ell-1}}{\overset{k, h_{0}, \dots, h_{\ell-1}}{\leftarrow}} & \underset{k, h_{0}, \dots, h_{\ell-1} \leftarrow^{\$} \mathbb{G}_{1} \\ & \underset{k, h_{0}, \dots, h_{\ell-1}}{\overset{k, h_{0}, \dots, h_{\ell-1}}{\leftarrow}} & \underset{k, h_{0}, \dots, h_{\ell-1} \leftarrow^{\$} \mathbb{G}_{1} \\ & \underset{k, h_{0}, \dots, h_{\ell-1}}{\overset{k, h_{0}, \dots, h_{\ell-1}}{\leftarrow}} & \underset{k, h_{0}, \dots, h_{\ell-1} \leftarrow^{\$} \mathbb{G}_{1} \\ & \underset{k, h_{0}, \dots, h_{\ell-1}}{\overset{k, h_{0}, \dots, h_{\ell-1}}{\leftarrow}} & \underset{k, h_{0}, \dots, h_{\ell-1} \leftarrow^{\$} \mathbb{G}_{1} \\ & \underset{k, h_{0}, \dots, h_{\ell-1}}{\overset{k, h_{0}, \dots, h_{\ell-1}}{\leftarrow}} & \underset{k, h_{0}, \dots, h_{\ell-1} \leftarrow^{\$} \mathbb{G}_{1} \\ & \underset{k, h_{0}, \dots, h_{\ell-1}}{\overset{k, h_{0}, \dots, h_{\ell-1}}{\leftarrow}} & \underset{k, h_{0}, \dots, h_{\ell-1} \leftarrow^{\$} \mathbb{G}_{1} \\ & \underset{k, h_{0}, \dots, h_{\ell-1} \leftarrow^{\$} \mathbb{G}_{\ell} \\ & \underset{k, h_{0}, \dots, h_{\ell-1} \leftarrow^{\flat} \mathbb{G}_{\ell} \\ & \underset{k, h_{0}, \dots, h_{\ell-1$

Fig. 10. Public coin interactive proof for \mathcal{R}_{LR} relation

Proposition 4. If the Discrete Logarithm Problem is hard in \mathbb{G}_1 , protocol 10 is a constant-round public coin argument of knowledge for $\mathcal{R}_{\mathsf{LR}}$ with perfect completeness, and perfect HVZK.

C Comparing UC and Game-Based Definition

C.1 Proof of Proposition 2

Proof. Let S be a simulator such that Π is indistinguishable from $\mathcal{F}_{SSLE} \circ S$ 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 $\mathcal{F}_{SSLE} \circ \mathcal{S}$.

Uniqueness: For any efficient \mathcal{A} corrupting |M| < t, observe that $\mathcal{A} \circ \mathcal{C}$ only requests an election to $\mathcal{F}_{\mathsf{SSLF}}$ when $|R \cap M| < \vartheta(|R|)$ by the way we defined the challenger.

Next we claim that, calling E the set used by $\mathcal{F}_{\mathsf{SSLE}}$ to record past elections (see Section 3), $(eid, i), (eid, j) \in E$ implies i = j. Assume by contradiction that $i \neq j$ and suppose (eid, i) is added first. Then (eid, j) 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 SSLE.Vrf_{pp}(*eid*, π, i) = SSLE.Vrf_{pp}(*eid*, π, j) = 1 and let $k \in [N] \setminus M$. As a consequence P_k receives from \mathcal{F}_{SSLE} both (result, *eid*, *i*) and (result, *eid*, *j*). By construction this implies (*eid*, *i*), (*eid*, *j*) $\in E$ and in particular i = j.

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

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

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 n, τ be as before when \mathcal{A} sends chall. In this phase, calling (eid, j) 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 index guessed by \mathcal{A}

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

C.2 SSLE schemes from Parametrised SSLE

In this section we show that a protocol Π realising $\mathcal{F}_{\mathsf{SSLE}}^{\kappa,\eta}$ can be compiled to an SSLE scheme with relaxed security guarantees. To this aim we still refer to the construction detailed in Figure 3.

Proposition 5. Given a protocol Π that (t, ϑ) -threshold realise $\mathcal{F}_{\mathsf{SSLE}}^{\kappa,\eta}$, the derived SSLE scheme defined in Figure 3 satisfy (t, ϑ) -threshold uniqueness, (t, ϑ) -threshold $(2^{-\kappa}+2^{-\eta})$ -fairness and (t, ϑ) -threshold $\xi(\kappa)$ -unpredictability with

$$\xi(\kappa) = \sup_{n \in \mathbb{N}} \left(\frac{n}{n - \vartheta(n)} \right) \frac{1}{2^{\kappa}} \frac{2^{\eta}}{2^{\eta} - 1}.$$

Proof. Let S be a simulator such that Π is indistinguishable from $S \circ \mathcal{F}_{\mathsf{SSLE}}^{\kappa,\eta}$ for any PPT environment in the class described in Definition 9. We first observe 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. This happens because the challenger \mathcal{C} only request an election if, calling R the currently registered users, $|R \cap M| < \vartheta(n)$. Next we prove that each of these properties holds replacing Π with $S \circ \mathcal{F}_{\mathsf{SSLE}}^{\kappa,\eta}$.

Uniqueness: Follows as in the proof of Proposition 2.

Fairness: Call R with n = |R| and $\tau = |R \cap M|$ the set of registered users when \mathcal{A} send chall and $\tau < \vartheta(n)$. Let **c** be the event that $\mathcal{F}_{\mathsf{SSLE}}^{\kappa,\eta}$ sends (corrupted, *eid*) to \mathcal{S} and **e** the event that j is set to \bot in the last election. By construction $\alpha \coloneqq \Pr[\mathsf{c}] \leq 2^{-\kappa}$ and $\beta \coloneqq \Pr[\mathsf{e}|\neg\mathsf{c}] \leq 2^{-\eta}$. Let also (*eid*, *j*) be the last value $\mathcal{F}_{\mathsf{SSLE}}^{\kappa,\eta}$ adds to his list after this election.

be the last value $\mathcal{F}_{\mathsf{SSLE}}^{\kappa,\eta}$ adds to his list after this election. If $\neg \mathbf{c}$ and $\neg \mathbf{e}$ then $j \sim U(R)$ by definition and P_j is the only winner, i.e. the only one for which $\mathcal{F}_{\mathsf{SSLE}}^{\kappa,\eta}$ broadcasts (result, *eid*, *j*). Therefore, calling for notational simplicity *E* the event $\mathsf{Exp}_{\mathsf{Fair}}^{\mathcal{A}}(1^{\lambda}, N, \vartheta) = 1$

$$\begin{aligned} \Pr\left[E\right] &= \Pr\left[E|\mathsf{c}\right] \Pr\left[\mathsf{c}\right] + \Pr\left[E|\neg\mathsf{c}\right] \Pr\left[\neg\mathsf{c}\right] \\ &= \alpha \Pr\left[E|\mathsf{c}\right] + (1-\alpha) \Pr\left[E|\neg\mathsf{c},\neg\mathsf{e}\right] \Pr\left[\neg\mathsf{e}|\neg\mathsf{c}\right] \\ &= \alpha \Pr\left[E|\mathsf{c}\right] + (1-\alpha)(1-\beta) \Pr\left[j \in R \setminus M|\neg\mathsf{c},\neg\mathsf{e}\right] \\ &= \alpha \Pr\left[E|\mathsf{c}\right] + (1-\alpha)(1-\beta) \left(\frac{n-\tau}{n}\right) \end{aligned}$$

Where in the second equation we used the fact that conditioning to **e** the event *E* never occurs. We can so conclude that $\left|\Pr\left[\mathsf{Exp}_{\mathsf{Fair}}^{\mathcal{A}}(1^{\lambda}, N, \vartheta) = 1\right] - \frac{n-\tau}{n}\right| \leq$

$$\begin{split} &\leq \left| \alpha \Pr\left[E|\mathbf{c}\right] + (1-\alpha)\frac{n-\tau}{n} + \beta(1-\alpha)\frac{n-\tau}{n} - \frac{n-\tau}{n} \right| \\ &\leq \alpha \left| \Pr\left[E|\mathbf{c}\right] - \frac{n-\tau}{n} \right| + \beta(1-\alpha)\frac{n-\tau}{n} \\ &\leq \alpha \max\left(\frac{\tau}{n}, \frac{n-\tau}{n}\right) + \beta \ \leq \ \frac{1}{2^{\kappa}} + \frac{1}{2^{\eta}}. \end{split}$$

Unpredictability: As before let R with n = |R| and $\tau = |R \cap M|$ be the set of registered users when \mathcal{A} send chall and $\tau < \vartheta(n)$, $H = R \setminus M$ the set of honest registered users, bad the event that $\mathcal{F}_{\mathsf{SSLE}}^{\kappa,\eta}$ sends (corrupted, *eid*) to \mathcal{S} and (*eid*, *j*) the last message $\mathcal{F}_{\mathsf{SSLE}}^{\kappa,\eta}$ store in his list E. Clearly there is a honest winner, event denoted by hw, if and only if $j \in R \setminus M$. Keeping the same notation used in the Fairness proof for c, e, α and β , we begin recalling that

$$\Pr\left[\mathsf{hw}\right] = \Pr\left[\mathsf{hw}|\mathsf{c}\right]\alpha + (1-\alpha)(1-\beta)\frac{n-\tau}{n}.$$

Next we upper bound the advantage of a given adversary, denoting with j' the index it returns at the end of the experiment

$$\begin{split} &\Pr\left[\mathsf{Exp}_{\mathsf{Unpr}}^{\mathcal{A}}(1^{\lambda},N,\vartheta)=1\right] \\ &=\Pr\left[j=j'|\mathsf{hw},\mathsf{\neg c}\right]\Pr\left[\mathsf{\neg c}|\mathsf{hw}\right]+\Pr\left[j=j'|\mathsf{hw},\mathsf{c}\right]\Pr\left[\mathsf{c}|\mathsf{hw}\right] \\ &\leq \frac{1}{n-\tau}\Pr\left[\mathsf{\neg c}|\mathsf{hw}\right]+\Pr\left[\mathsf{c}|\mathsf{hw}\right] \\ &= \frac{1}{n-\tau}+\frac{n-\tau-1}{n-\tau}\Pr\left[\mathsf{c}|\mathsf{hw}\right] \,\leq \, \frac{1}{n-\tau}+\Pr\left[\mathsf{c}|\mathsf{hw}\right]. \end{split}$$

To conclude we then only need to upper bound $\Pr[c|hw]$

$$\begin{aligned} \Pr\left[\mathsf{c}|\mathsf{hw}\right] &= \frac{\Pr\left[\mathsf{hw}|\mathsf{c}\right] \cdot \Pr\left[\mathsf{c}\right]}{\Pr\left[\mathsf{hw}\right]} \\ &= \alpha \Pr\left[\mathsf{hw}|\mathsf{c}\right] \cdot \left(\Pr\left[\mathsf{hw}|\mathsf{c}\right]\alpha + (1-\alpha)(1-\beta)\frac{n-\tau}{n}\right)^{-1} \\ &\leq \alpha \cdot \left(\alpha + (1-\alpha)(1-\beta)\frac{n-\tau}{n}\right)^{-1} \\ &\leq \frac{1}{2^{\kappa}} \left(\frac{1}{2^{\kappa}} + (1-2^{-\kappa})(1-2^{-\eta})\frac{n-\tau}{n}\right)^{-1} \\ &= \frac{n}{n+(2^{\kappa}-1)(1-2^{-\eta})(n-\tau)} \\ &\leq \frac{1}{2^{\kappa}} \cdot \frac{2^{\eta}}{2^{\eta}-1} \cdot \frac{n}{n-\tau} \\ &\leq \frac{1}{2^{\kappa}} \cdot \frac{2^{\eta}}{2^{\eta}-1} \cdot \frac{n}{n-\vartheta(n)} \leq \frac{1}{2^{\kappa}} \cdot \frac{2^{\eta}}{2^{\eta}-1} \cdot \sup_{n\in\mathbb{N}} \left(\frac{n}{n-\vartheta(n)}\right) \end{aligned}$$

Where in the first inequality we used the fact that the given function is monotone in $\Pr[\mathsf{hw}|\mathsf{c}] \in [0, 1]$. Similarly we used the same argument in the second inequality with $\alpha \in [0, 2^{-\kappa}]$ and $\beta \in [0, 2^{-\eta}]$.

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

$$c^{(1)} = \left([\mathbf{k}]_1, \left[\sigma x_{b,1} + \mathbf{k}^\top \mathbf{w}_1 \right]_1, \dots, \left[\sigma x_{b,n} + \mathbf{k}^\top \mathbf{w}_n \right]_1 \right)$$

- 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 $\mathbf{w}_i^* \sim U(\mathbb{F}_q^2)$

$$\begin{split} \mathsf{mpk}^{(2)} &= \left(\left[\mathbf{a} \right]_1, \left[\mathbf{a}^\top \mathbf{w}_1^* \right]_1, \dots, \left[\mathbf{a}^\top \mathbf{w}_n^* \right]_1 \right) \\ c^{(2)} &= \left(\left[\mathbf{k} \right]_1, \left[\mathbf{k}^\top \mathbf{w}_1^* \right]_1, \dots, \left[\mathbf{k}^\top \mathbf{w}_n^* \right]_1 \right) \\ \mathsf{sk}_{\mathbf{y}}^{(2)} &= \left(\left[\sum_{i=1}^n ry_i \mathbf{w}_i - r\sigma(\mathbf{x}_b^\top \mathbf{y}) \cdot \widehat{\mathbf{a}} \right]_2, \left[r \right]_2 \right) \end{split}$$

 $- \mathsf{H}_3^b$: As H_2^b but each $\mathsf{sk}_{\mathbf{y}}$ is generated by sampling a fresh $\delta \leftarrow \mathbb{F}_q$ and returning

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

The thesis follows if H_0^0 is indistinguishable from H_0^1 . To this aim we argue that H_0^b , H_1^b cannot be distinguished if DDH is hard in \mathbb{G}_1 , H_1^b , H_2^b are statistically close, H_2^b , H_3^b are indistinguishable assuming DDH is hard over \mathbb{G}_2 and H_2^0 , H_1^1 are equally distributed.

 $- H_0^b - H_1^b$. 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 $\rho \leftarrow^{\$} \mathbb{F}_{q}$ and set $[\mathbf{a}]_{1} \leftarrow [(\rho, \rho\alpha)]_{1}, [\mathbf{k}]_{1} \leftarrow [(\beta, \gamma)]_{1}$ 2: Sample $\mathbf{w}_{1}, \dots, \mathbf{w}_{n} \leftarrow^{\$} \mathbb{F}_{q}^{2}$ and run $(\mathbf{x}_{0}, \mathbf{x}_{1}) \leftarrow^{\$} \mathcal{D}(1^{\lambda}, n)$ 3: Compute $\mathsf{mpk} \leftarrow ([\mathbf{a}]_{1}, [\mathbf{a}^{\top}\mathbf{w}_{1}]_{1}, \dots, [\mathbf{a}^{\top}\mathbf{w}_{n}]_{1})$ 4: $\sigma \leftarrow^{\$} \mathbb{F}_{q} \setminus \{0\}, \quad c \leftarrow ([\mathbf{k}]_{1}, [\sigma x_{b,1} + \mathbf{k}^{\top}\mathbf{w}_{1}]_{1}, \dots, [\sigma x_{b,n} + \mathbf{k}^{\top}\mathbf{w}_{n}]_{1})$ 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 $\mathcal{D} \leftarrow \bot$. Else: 7: Compute $\mathsf{sk}_{\mathbf{y}} \leftarrow [\sum_{i=1}^{n} ry_{i}\mathbf{w}_{i}]_{2}, [r]_{2}$ and send $\mathcal{D} \leftarrow \mathsf{sk}_{\mathbf{y}}$ 8: When $b' \leftarrow \mathcal{D}$: Return b'.

Recall than in DDH¹ calling $s = \beta \rho^{-1} \sim U(\mathbb{F}_q)$ then $\mathbf{k} = s\mathbf{a}$, while \mathbf{a}, \mathbf{k} are uniform and independent in DDH⁰. 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 , **a**, **k** are uniform over \mathbb{F}_q^2 , hence the probability that **k** lies in the linear span of **a** is q^{-1} . Up to this negligible probability **a**, **k** are independent, so there exists a unique vector $\hat{\mathbf{a}}$ satisfying $\mathbf{a}^{\top}\hat{\mathbf{a}} = 0$ and $\mathbf{k}^{\top}\hat{\mathbf{a}} = 1$. Calling $\mathbf{w}_i^* = \mathbf{w}_i + \sigma x_{b,i} \cdot \hat{\mathbf{a}}$ this is still uniformly distributed and we obtain the distribution of game H_2^b since

$$\mathbf{a}^{\top}\mathbf{w}_{i} = \mathbf{a}^{\top}(\mathbf{w}_{i}^{*} - \sigma x_{b,i} \cdot \widehat{\mathbf{a}}) = \mathbf{a}^{\top}\mathbf{w}_{i}^{*}$$
$$\sigma x_{b,i} + \mathbf{k}^{\top}\mathbf{w}_{i} = \sigma x_{b,i} + \mathbf{k}^{\top}(\mathbf{w}_{i}^{*} - \sigma x_{b,i} \cdot \widehat{\mathbf{a}}) = \mathbf{k}^{\top}\mathbf{w}_{i}^{*}$$
$$\sum_{i=1}^{n} y_{i}\mathbf{w}_{i} = \sum_{i=1}^{n} y_{i}(\mathbf{w}_{i}^{*} - \sigma x_{b,i} \cdot \widehat{\mathbf{a}}) = \sum_{i=1}^{n} y_{i}\mathbf{w}_{i}^{*} - \sigma(\mathbf{x}_{b}^{\top}\mathbf{y}) \cdot \widehat{\mathbf{a}}.$$

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

 $-H_2^b-H_3^b$. 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 $\mathbf{a}, \mathbf{k}, \mathbf{w}_1, \dots, \mathbf{w}_n \leftarrow^{\$} \mathbb{F}_q^2$ and run $\mathcal{D}(1^{\lambda}, n) \to (\mathbf{x}_0, \mathbf{x}_1)$ 2: Compute $\hat{\mathbf{a}}$ such that $\mathbf{k}^{\top} \hat{\mathbf{a}} = 1$ and $\mathbf{a}^{\top} \hat{\mathbf{a}} = 0$ 3: Set $\mathsf{mpk} \leftarrow ([\mathbf{a}]_1, [\mathbf{a}^{\top} \mathbf{w}_1]_1, \dots, [\mathbf{a}^{\top} \mathbf{w}_n]_1)$ 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 $\mathcal{D} \leftarrow \bot$. Else: 6: Set $\mathsf{sk}_{\mathbf{y}} \leftarrow [\sum_{i=1}^n r_j y_i \mathbf{w}_i - \delta_j (\mathbf{x}_b^{\top} \mathbf{y}) \cdot \hat{\mathbf{a}}]_2, [r_j]_2$ 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 $\hat{\mathbf{a}}$ can be computed. Thus when \mathcal{A} receives a truly random tuple in DDH^0_ℓ it perfectly simulates H^b_3 . Conversely $\delta_j = \sigma r_j$ in DDH^1_ℓ , where $\sigma \neq 0$ with overwhelming probability, implies that \mathcal{A} sends message distributed as in H^b_2 . 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 $[\sum_{i=1}^n ry_i \mathbf{w}_i]_2$, $[r]_2$. 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 of Theorem 1. To prove the statement we have to provide a simulator S interacting with $\mathcal{F}_{SSLE}^{\lambda,\eta}$ such that for all Z statically corrupting T < N players¹², $S \circ \mathcal{F}_{SSLE}$ is indistinguishable from the real protocol.

In the following, given $R \subseteq \mathbb{F}_q^2$ we call $R_{|j} = \{\gamma \in \mathbb{F}_q : (j, \gamma) \in R\}.$

Decryption of S: Initially set $S, R \leftarrow \emptyset$, $n \leftarrow 0$, mpk, msk $\leftarrow^{\$}$ MKS.Setup (1^{λ}) and wait for $M \leftarrow^{\$} \mathcal{Z}$. Upon receiving

- setup from P_j : send (input, mpk).
- keygen from P_j : When there is no pending election, increment $n \leftarrow n+1$ until $n \notin S$. Add $R \leftarrow R \cup \{(j,n)\}$ and generate $\mathsf{sk}_n \leftarrow \mathsf{MKS}.\mathsf{KeyGen}(n,\mathsf{msk})$., send (key, sk_n) to P_j and broadcast (key requested, i, n).

 $^{^{12}}$ if $\mathcal Z$ corrupts all players the simulation becomes trivial

- (registered, j) from $\mathcal{F}_{SSLE}^{\lambda,\eta}$: When there is no pending election, increment $n \leftarrow n+1$ until $n \notin S$. Add $R \leftarrow R \cup \{(j, n)\}$ and broadcast (key_requested).
- (electing, *eid*) from $\mathcal{F}_{\mathsf{SSLE}}^{\lambda,\eta}$: Set $p = |S \cap [n]| \cdot n^{-1}$ and send (prob, *eid*, 0, p) to $\mathcal{F}_{\mathsf{SSLE}}^{\lambda,\eta}$.
- A request to send (outcome, eid) from $\mathcal{F}_{SSLE}^{\lambda,\eta}$: If a malicious user P_i would receive (outcome, eid, 1), sample $\gamma \leftarrow R_{i}, m \leftarrow [\kappa n]$ such that $m = \gamma \mod n$ and compute $c \leftarrow$ ^{\$} MKS.Enc((m, n), mpk). Else set $c \leftarrow$ ^{\$} MKS.Enc($(2^{\lambda}, n)$, mpk). Broadcast (challenge, *eid*, *c*).
- (reject, eid, j) from $\mathcal{F}_{\mathsf{SSLE}}^{\lambda,\eta}$ with P_j uncorrupted: Set $(\gamma, \pi) \leftarrow \bot$. Broadcast (claim, eid, γ, π) as P_j . (result, eid, j) from $\mathcal{F}_{\mathsf{SSLE}}^{\lambda,\eta}$ with P_j uncorrupted: Sample $\gamma \leftarrow \ R_{|j}$ and simulate π . Broadcast $(\mathsf{claim}, eid, \gamma, \pi)$ as P_i .
- (claim, eid, γ, π) from P_j : Verify π . If π is correct, send (reveal, eid) to $\mathcal{F}_{SSLE}^{\lambda, \eta}$. Otherwise send (fake reject, *eid*).

Hybrid Games: Given a PPT environment performing at most L elections, we define a sequence of hybrid functionalities.

- H_{real} : The real protocol.
- H₀: as H_{real} but all NIZK proofs are simulated.
- H₁: as H₀ initially set $E \leftarrow \emptyset$. For each election with index *eid*, calling *m* the element sampled by \mathcal{F}_{SnC} , set $\gamma = m \mod n$, retrieve $(j, \gamma) \in R$, store $E \leftarrow E \cup \{(eid, j)\}$ and let honest user P_j return (outcome, eid, 1) if $(\gamma, j) \in R$ or (outcome, eid, 0) otherwise.
- $-H_2$: as H_1 but when \mathcal{Z} send (reveal, *eid*) to an honest user P_i find (*eid*, *j*) $\in E$. If i = j, set $\gamma \leftarrow m \mod n$, with m being the message generate during the election *eid*, and simulate π . Else, set $(\gamma, \pi) \leftarrow \bot$. Broadcast (claim, *eid*, γ, π) from P_i .
- $-H_3$: as H_2 but when a corrupted P_j sends (claim, eid, γ, π), if $(j, \gamma) \in R, \pi$ is accepting and $(eid, j) \in E$ then all honest users return (result, eid, j). Else they return (rejected, eid, j).
- $-\mathsf{H}_4^{\ell}$: as H_3 but for the first ℓ elections, if (eid, γ) is added in E with $(j, \gamma) \notin R$ for all malicious users' index $j \in M$, then set

$$c \leftarrow^{\$} \mathsf{MKS}.\mathsf{Enc}((2^{\lambda}, 0), \mathsf{mpk}).$$

Claim 1. $H_{real} \equiv H_0$: Follows from perfect zero-knowledge of the argument used, which implies that the two functionalities are identically distributed.

Claim 2. $H_0 \equiv H_1$: Follows as the only difference in the two game is the outcome message of honest players after an election occurs. Their behaviour is identically though because in H_1 , if P_i returns (outcome, eid, 1), then (eid, i) is added to E and $m = \gamma + \delta n$, with $(i, \gamma) \in R$.

By construction $(i, \gamma) \in R$ implies that P_i received the secret key sk_{γ} . By the correctness of the Modular PEKS scheme then, as c encrypts the message $m = \gamma + \delta n$ for some $\delta \in [\kappa]$

$$\mathsf{MKS}.\mathsf{Dec}(c,\mathsf{mpk},\mathsf{sk}_{\gamma})\to 1$$

which implies that P_i would return (outcome, eid, 1) also in H_0 . A similar argument shows that P_i returns (outcome, eid, 0) in H_1 if it would return it in H_0 .

Claim 3. $H_1 \equiv H_2$: The only difference in this two games is how honest players replies to the input (reaveal, *eid*).

In H₂, an uncorrupted user P_i replies with (claim, eid, γ, π) with simulated π if and only if $(eid, i) \in E$ and in particular $m = \gamma + \delta n$ for some $\delta \in [\kappa]$. As in the previous claim this means that $(i, \gamma) \in R$, so P_i received the key sk_{γ} , which decrypts the ciphertext c to 1.

An analogous argument show that the two functionality behaves identically also when P_i would return (claim, eid, γ, π) with $(\gamma, \pi) = \bot$.

Claim 4. $H_2 \equiv H_3$: The difference in this case lies on how the honest parties behaves when a corrupted P_j sends (claim, eid, γ, π). In particular in H_3 the extra check $(eid, j) \in E$ is performed, hence we only need to show that if honest parties returns (result, eid, j) in H_2 then the same happens in H_3 .

Recall that by construction, this condition on H₂ means that π is accepted by NIZK.V_{Dec}. Calling c the challenge ciphertext for election *eid*, using simulation soundness and the fact that proofs are made unique (see Section 2.7), the correctness of π implies that there exists a secret key $\mathsf{sk}_{\gamma} \in \mathcal{L}_{\mathsf{key}}$ such that

MKS.Dec
$$(c, \mathsf{mpk}, \mathsf{sk}_{\gamma}) \to 1$$
.

Since c is the encryption of some (m, n) with $m \in [\kappa n]$, this implies that $m = \gamma \mod n$. Finally, as $(i, \gamma) \in R$, this means that $(eid, i) \in E$ by construction.

Claim 5. $H_4^{\ell} \equiv H_4^{\ell-1}$: This time we prove the two functionality are only computationally indistinguishable by transforming any PPT distinguisher \mathcal{D} into an attacker \mathcal{A} for the selective security of the underlying Modular PEKS scheme.

Description of \mathcal{A} : Call p an upper bound on the number of registration performed by \mathcal{D} and sample $n^* \leftarrow^{\$} [p + \kappa L], m^* \leftarrow^{\$} [\kappa L]$ and set $x_0 = (2^{\lambda}, 0), x_1 = (m^*, n^*)$. Send (x_0, x_1) to the challenger. Upon receiving (mpk, c^*), simulate $\mathsf{H}_4^{\ell-1}$ with the following modifications:

- When a malicious user P_j register, query to the challenger sk_n
- During the ℓ -th election, calling $\gamma^* \leftarrow m^* \mod n^*$, check that

$$n = n^*, \quad \forall j \in M, \ (j, \gamma^*) \notin R.$$

If any of these conditions does not hold, return a random bit. If $(j, \gamma^*) \in R$ then add $E \leftarrow E \cup \{(eid, j)\}$ and resume the execution of $\mathsf{H}_4^{\ell-1}$.

Finally, when $\mathcal{D} \to b$, return the same bit.

Proof of Claim. To begin with we estimate the probability that the check performed at the ℓ -th election passes – we call this event check. Observe that during each election the size of S grows at most by κ and that, calling n_0 the number of registered user at a given time, then $n \leq n_0 + |S| \leq n_0 + \kappa L \leq p + \kappa L$. As the simulation does not depend on n^* until the ℓ -th election, n and n^* are independent. So $\Pr[n = n^*] = (p + \kappa L)^{-1}$. Next, conditioning on $n = n^*$ and calling hw the

probability that no dishonest player would win the ℓ -th election, we have that

$$\Pr\left[\nexists j \in M \ : \ (j,\gamma^*) \in R | n^* = n\right] \ = \ \Pr\left[\mathsf{hw}\right]$$

Therefore the check passes with probability

$$\Pr\left[\mathsf{check}\right] \geq \Pr\left[n=n^*\right] \cdot \Pr\left[\nexists j \in M : (j,\gamma^*) \in R | n=n^*\right] \geq \frac{1}{p+\kappa L} \cdot \Pr\left[\mathsf{hw}\right]$$

which is significant if hw happens with significant probability.

Next we show that conditioning on check, \mathcal{A} never query a key that applied on x_0 and x_1 would yields different values (in particular, every key will decrypt both to 0).

Indeed for any key associated to a value γ requested by \mathcal{A} for a corrupted user P_j , $\gamma \leq p + \kappa L$ is polynomially small. This means that $\gamma + \delta \cdot 0 = \gamma$ will never be equal to 2^{λ} .

On the other side, call $m^* = \gamma^* + \delta^* n^*$. If P_j requests a key for γ before the ℓ -th election with $m^* = \gamma + \delta n^*$ with $\delta \in [\kappa]$, then

$$\gamma + \delta n^* = \gamma^* + \delta^* n^* \quad \Rightarrow \quad \gamma - \gamma^* = (\delta^* - \delta) n^*$$

As both γ and γ^* are smaller than n^* this may only happen if $\gamma = \gamma^*$, which implies $(j, \gamma^*) = (j, \gamma) \in \mathbb{R}$, contradicting our assumption on check.

Conversely, if this key is requested after the i-th step, then by the previous equation

$$\gamma = \gamma^* + (\delta^* - \delta)n^* \in S$$

as $(\delta^* - \delta) \in [\kappa]$ which is again a contradiction.

We briefly observe now that when c^* encrypts x_0 , \mathcal{A} perfectly simulates, upon conditioning on check, the functionality $\mathsf{H}_4^{\ell-1}$ conditioned to the even hw that no dishonest player wins the ℓ -th election. Conversely, when the challenger encrypts x_1 , it perfectly simulates H_4^{ℓ} again conditioned to hw. Hence, calling b' the challenger's bit

$$\begin{split} \mathsf{Adv}_{\mathsf{SSFE}}^{\mathcal{A}}(1^{\lambda}) &= \left| \Pr\left[\mathcal{A} \to b'\right] - \frac{1}{2} \right| \\ &= \left| \Pr\left[\mathcal{A} \to b' |\mathsf{check}\right] \Pr\left[\mathsf{check}\right] + \Pr\left[\mathcal{A} \to b' |\neg\mathsf{check}\right] \Pr\left[\neg\mathsf{check}\right] - \frac{1}{2} \right| \\ &= \left| \Pr\left[\mathcal{D} \to b' |\mathsf{hw}\right] \Pr\left[\mathsf{check}\right] + \frac{1}{2} \Pr\left[\neg\mathsf{check}\right] - \frac{1}{2} \right| \\ &= \left| \Pr\left[\mathcal{D} \to b' |\mathsf{hw}\right] - \frac{1}{2} \right| \cdot \Pr\left[\mathsf{check}\right] \\ &\geq \left| \Pr\left[\mathcal{D} \to b' |\mathsf{hw}\right] - \frac{1}{2} \right| \cdot \frac{1}{p + \kappa L} \cdot \Pr\left[\mathsf{hw}\right]. \end{split}$$

In order to conclude we remark that \mathcal{D} only has a chance to win if hw occurs, as otherwise the functionality H_4^ℓ and $H_4^{\ell-1}$ are indistinguishable. This means that

$$\begin{aligned} \mathsf{Adv}(\mathcal{D}) &= \left| \Pr\left[\mathcal{D} \to b'\right] - \frac{1}{2} \right| \\ &= \left| \Pr\left[\mathcal{D} \to b' |\mathsf{hw}\right] \Pr\left[\mathsf{hw}\right] + \Pr\left[\mathcal{D} \to b' |\neg\mathsf{hw}\right] \Pr\left[\neg\mathsf{hw}\right] - \frac{1}{2} \right| \\ &= \left| \Pr\left[\mathcal{D} \to b'\right] - \frac{1}{2} \right| \cdot \Pr\left[\mathsf{hw}\right] \end{aligned}$$

This allow us to conclude that $\mathsf{Adv}_{\mathsf{SSFE}}^{\mathcal{A}}(1^{\lambda}) \cdot (p + \kappa L) \geq \mathsf{Adv}(\mathcal{D})$, that is therefore negligible.

D.3 Proof of Theorem 2

Proof of Theorem 2. In order to prove the statement we must provide a simulator S that interacts with $\mathcal{F}_{\mathsf{SSLE}}^{\kappa,\eta}$ such that the protocol is indistinguishable from $S \circ \mathcal{F}_{\mathsf{SSLE}}^{\kappa,\eta}$ for any PPT environment Z that statically corrupts strictly less than t players. Intuitively we use the security of Threshold Elgamal to make the simulator alter the first component of the message contained in the ℓ -th ciphertext. In particular this is fixed to -1 when an honest user wins as the resulting ciphertext cannot be decrypted by anyone else. By selective security of the underlying FE scheme 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 S is provided below, where we omit to specify the behaviour of honest parties when this equals the correct one. We remark however that S can always simulate it as he initially generates all the public and private parameters. To simply notation we also assume that the number of corrupted parties |M| = t - 1 and denote

$$R_{|j} = \{\gamma : (j,\gamma) \in R\}$$

the set of values in [n] associated to player P_j .

Description of S: Initially set $B, R, S \leftarrow \emptyset, n \leftarrow 0$, wait M from \mathcal{Z} and forward it to $\mathcal{F}_{\mathsf{SSLE}}^{\kappa,\eta}$. Generate $\mathsf{mpk}, \mathsf{msk} \leftarrow^{\$} \mathsf{FE}.\mathsf{Setup}(1^{\lambda}, 3)$ and sample $f \leftarrow^{\$} \mathbb{F}_q[x]_{\leq t}, g \leftarrow^{\$} \mathbb{G}_1$. Call $x \leftarrow f(-1)$ and compute $h \leftarrow g^x, k_i \leftarrow g^{f(i)}, \mathsf{pp} \leftarrow (\mathsf{mpk}, g, h, k_0, \ldots, k_{N-1})$. Upon receiving

- (setup) from P_j : send (input, pp, f(j)) to P_j .
- (keygen) from P_{ν} : Wait for all elected leader to reveal themselves. While $n \in S$, increment n by one. Compute $\mathsf{sk}_{n,\delta} \leftarrow^{\$} \mathsf{FE}.\mathsf{KeyGen}(\mathbf{y}_{n,\delta},\mathsf{msk})$, update $R \leftarrow R \cup \{(\nu, n)\}$ and $n \leftarrow n + 1$. Set $\mathsf{sk}_n \leftarrow \{\mathsf{sk}_{n,\delta}\}_{\delta \in [\kappa]}$, send $P_{\nu} \leftarrow (\mathsf{key},\mathsf{sk}_n)$ and broadcast (key_requested, ν) to dishonest users. Send (register) to $\mathcal{F}^{\kappa,\eta}_{\mathsf{SSLF}}$ as P_{ν} .
- (electing, *eid*) from $\mathcal{F}_{SSLE}^{\kappa,\eta}$: let *p* be the probability that a random subset *Q* of registered players is contained in *M* and send (prob, *eid*, *p*).
- A request to send (outcome, eid) from \$\mathcal{F}_{SSLE}^{\kappa,\eta}\$ without receiving (corrupted, eid) first: Sample \$\gamma^* \leftarrow \$\binom{1}{1}\$ [n] and, if \$\gamma^* \in S\$ set \$\gamma \leftarrow \$\gamma^*\$. Conversely read the messages sent to dishonest users. If \$P_j\$ received (outcome, eid, 1) for some \$j \in M\$, set \$\gamma \leftarrow \$\binom{8}{1}\$, otherwise set \$\gamma \leftarrow \$\pm\]. Sample \$G, \$H\$ \leftarrow \$\binom{8}{1}\$, \$Q\$ a subset of the registered players not contained in \$M\$ and call \$i\$ = min(\$Q \ M\$).
 Simulate honestly all player but \$P_i\$. For \$P_i\$ compute

$$u_{1}, v_{1}, u_{2}, v_{2} \leftarrow^{\$} \mathbb{G}_{1}, \qquad s_{i}, \sigma \leftarrow^{\$} \mathbb{F}_{q}$$
$$\mathbf{c}_{0,i} \leftarrow [s_{i}\mathbf{a}]_{1}, \quad c_{1,i} \leftarrow v_{1}, \quad c_{2,i} \leftarrow v_{2}^{-1} \cdot \left[-\sigma + s_{i}\mathbf{a}^{\top}\mathbf{w}_{2}\right]_{1}$$
$$c_{3,i} \leftarrow v_{2}^{-n} \cdot \left[-n\sigma + s_{i}\mathbf{a}^{\top}\mathbf{w}_{3}\right]_{1}, \quad G_{1} \leftarrow u_{1}, \quad G_{2} \leftarrow u_{2}.$$

Simulate $\pi_{\mathsf{LR},i}$ and broadcast as $\mathcal{F}_{\mathsf{CT}}^{\mathsf{ch}}$ (tossed, eid, G, H, Q) and as P_i the tuple

$$(msg, eid, \mathbf{c}_{0,i}, c_{1,i}, c_{2,i}, c_{3,i}, G_{1,i}, G_{2,i}, \pi_{\mathsf{LR},i})$$

• Upon receiving $(\mathsf{msg}, eid, \mathbf{c}_{0,\mu}, c_{1,\mu}, c_{2,\mu}, c_{3,\mu}, G_{1,\mu}, G_{2,\mu}, \pi_{\mathsf{LR},\mu})$ from dishonest P_{μ} : Let Q_0 be the set of $\mu \in Q$ such that $\pi_{\mathsf{LR},\mu}$ is accepted. For all $\mu \in Q_0$ extract by brute force m_{μ} as the discrete logarithm in base HG^{-x} of

$$c_{1,\mu} \cdot G_{1,\mu}^{-x} \cdot \mathbf{c}_{0,\mu}^{-\mathbf{w}_1}$$

If $\gamma = \perp$ set m = -1 otherwise find the only $m_i \in [n]$ such that

$$\widehat{m} = \sum_{\mu \in Q_0 \setminus \{i\}} m_i \qquad \widehat{m} + m_i = \gamma \mod n$$

and call $m = \hat{m} + m_i$

$$K_{1} \leftarrow \left[\sigma m + s_{i} \mathbf{a}^{\top} \mathbf{w}_{1}\right]_{1} \cdot \left(v_{1} \cdot \prod_{\mu \in Q_{0} \setminus \{i\}} G_{1,\mu}^{x} G^{-xm_{\mu}} H^{m_{\mu}}\right)^{-1}$$
$$K_{2} \leftarrow \left(v_{2} \cdot \prod_{\mu \in Q_{0} \setminus \{i\}} G_{2,\mu}^{x}\right)^{-1}$$

Add $B \leftarrow B \cup \{(eid, K_1, K_2, \widehat{m}, n, R)\}$, compute $G_1 \leftarrow \prod_{\mu \in Q_0} G_{1,i}$ and $G_2 \leftarrow G \cdot \prod_{\mu \in Q_0} G_{2,\mu}$. Produce for all $\nu \in M$ the associated group share $K_{1,\nu} \leftarrow G_1^{f(\nu)}$ and $K_{2,\nu} \leftarrow G_2^{f(\nu)}$ and for $j \notin M$ the fake decryption share

$$K_{1,j} \leftarrow K_1^{\lambda_{-1}} \cdot \prod_{\nu \in M} K_{1,\nu}^{\lambda_{\nu}}, \qquad K_{2,j} \leftarrow K_2^{\lambda_{-1}} \cdot \prod_{\nu \in M} K_{2,\nu}^{\lambda_{\nu}}$$

where λ_{ν} are the Lagrange coefficients associated to $M \cup \{-1\}$ to evaluate in j. Finally simulate $\pi_{\text{DDH},j}$ and broadcast (open, $eid, K_{1,i}, K_{2,j}, \pi_{\text{DDH},j}$).

If $\gamma \in S$ wait for users to repeat the election, otherwise allow $\mathcal{F}_{\mathsf{SSLE}}^{\kappa,\eta}$ to send the outcome, *eid* messages.

- (corrupted, eid) from $\mathcal{F}_{SSLE}^{\kappa,\eta}$:

Sample $G, H \leftarrow^{\$} \mathbb{G}_1, Q \subseteq M$ and broadcast (tossed, eid, G, H, Q). During the protocol extract as before m_{μ} for every $P_{\mu} \in Q_0$ with Q_0 the set of parties that send msg with a correct proof. Call $m = \sum_{\mu \in Q_0} m_{\mu}$ and $\gamma = m \mod n$.

- (result, eid, j) from $\mathcal{F}_{\mathsf{SSLE}}^{\kappa, \eta}$ and $j \notin M$:

Find $(eid, K_1, K_2, \widehat{m}, \widehat{n}, \widehat{R}) \in B$, sample $\gamma \leftarrow \widehat{R}_{|j}$, set $m_i \in [\widehat{n}]$ such that $m_i = \gamma - \widehat{m} \mod \widehat{n}$ and call $m \leftarrow \widehat{m} + m_i$. Next find δ such that $m = \delta \widehat{n} + \gamma$. Simulate π_{Dec} from NIZK.S_{Dec} and broadcast (claim, $eid, \pi, K_1, K_2, \gamma, \delta$).

Update $S \leftarrow S \cup \{\gamma + \delta'\hat{n} : \delta' \in [\kappa]\}$. - (claim, $eid, K_1, K_2, \pi, \gamma, \delta$) from P_{ν} : If $(\nu, \gamma) \in R$ and π is accepted, send (reveal, eid) to $\mathcal{F}_{\mathsf{SSLE}}^{\kappa,\eta}$ as P_{ν} . Otherwise send (fake_reject, eid, ν) to $\mathcal{F}_{\mathsf{SSLE}}^{\kappa,\eta}$.

Next, calling L the maximum number of elections \mathcal{Z} requests, we provide a sequence of hybrid functionalities lying between the real protocol and $\mathcal{F}_{SSLE}^{\kappa,\eta} \circ \mathcal{S}$. To avoid repetitions observe that the messages $G_{1,\mu}, G_{2,\mu}, \mathbf{c}_{0,\mu}, c_{1,\mu}$ sent by each member P_{μ} in the random committee with tag msg uniquely determine four field elements $m_{\mu}, r_{\mu}, s_{\mu}, \rho_{\mu}$ such that

$$G_{1,\mu} = g^{r_{\mu}} G^{m_{\mu}}, \quad G_{2,\mu} = g^{\rho_{\mu}}, \quad \mathbf{c}_{0,\mu} = [s_{\mu} \mathbf{a}]_{1}, \quad c_{1,\mu} = h^{r_{\mu}} H^{m_{\mu}} \left[s_{\mu} \mathbf{a}^{\top} \mathbf{w}_{1} \right]_{1}$$

We then define $\widehat{m}, \widehat{r}, \widehat{s}, \widehat{\rho}$ to be the sum of these elements for $\mu \in Q_0 \setminus \{i\}$, with $i = \min Q \setminus M$.

- H_{real}: the real protocol.
- H₀: All the NIZK proof sent by honest users are simulated.
- H_0^{\star} but in the threshold decryption compute $K_{1,j} \leftarrow G_{1,j}^{f(j)}$ and $K_{2,j} \leftarrow G_{1,j}^{f(j)}$ for $j \in M$. Next set $\widetilde{K}_1 \leftarrow G_1^x$ and $\widetilde{K}_2 \leftarrow G_2^x$ with x = f(-1) and for each honest user P_{ν} set the decryption shares as

$$K_{1,\nu} \leftarrow \widetilde{K}_1^{\lambda_{-1}} \prod_{j \in M} K_{1,j}, \quad K_{2,\nu} \leftarrow \widetilde{K}_2^{\lambda_{-1}} \prod_{j \in M} K_{2,j}$$

where λ_i are the Lagrange coefficients to evaluate in ν .

- $\begin{array}{l} \ \mathsf{H}_{1}^{0} \coloneqq \mathsf{H}_{0}^{\star}. \ \mathsf{H}_{1}^{\ell}: \text{ as } \ \mathsf{H}_{1}^{\ell-1} \text{ but in the } \ell\text{-th election } P_{i} \text{ samples randomly } u_{1}, v_{1} \leftarrow \$ \ \mathbb{G}_{1} \text{ and set } \\ G_{1,i} \leftarrow u_{1}G^{m_{i}} \text{ and } c_{1,i} \leftarrow v_{1}H^{m_{i}} \left[s_{i}\mathbf{a}^{\top}\mathbf{w}_{1}\right]_{1}. \\ \text{Moreover in the threshold decryption } \widetilde{K}_{1} \leftarrow v_{1}h^{\widehat{r}}G^{x(\widehat{m}+m_{i})}. \end{array}$
- $\begin{array}{l} \ \mathsf{H}_{2}^{0} \coloneqq \mathsf{H}_{1}^{L}. \ \mathsf{H}_{2}^{\ell}: \text{ as } \mathsf{H}_{2}^{\ell-1} \text{ but in the } \ell\text{-th election } P_{i} \text{ samples } u_{2}, v_{2} \leftarrow^{\$} \mathbb{G}_{1} \text{ and set } G_{2,i} \leftarrow u_{2}, \ c_{2,i} \leftarrow v_{2}^{-1} \left[s_{i} \mathbf{a}^{\top} \mathbf{w}_{2} \right]_{1}, \ c_{3,i} \leftarrow v_{2}^{-n} \left[s_{i} \mathbf{a}^{\top} \mathbf{w}_{3} \right]_{1}. \\ \text{Moreover in threshold decryption } \widetilde{K}_{2} \leftarrow v_{2} h^{\widehat{\rho}} G^{x}. \end{array}$
- H_3 : as H_2^L but, sampled $\theta, \sigma \leftarrow \mathbb{F}_q$ calling $G = g^{\theta}$ and $H = h^{\theta}[\sigma]_1, P_i$ computes

$$G \leftarrow g^{\theta}, \qquad H \leftarrow h^{\theta} [\sigma]_{1}, \qquad G_{1,i} \leftarrow u_{1}, \qquad G_{2,i} \leftarrow u_{2}, \qquad c_{1,i} \leftarrow v_{1}$$
$$c_{2,i} \leftarrow v_{2}^{-1} \left[-\sigma + s_{i} \mathbf{a}^{\top} \mathbf{w}_{2} \right]_{1}, \qquad c_{3,i} \leftarrow v_{2}^{-n} \left[-n\sigma + s_{i} \mathbf{a}^{\top} \mathbf{w}_{3} \right]_{1}.$$

Moreover for threshold decryption, calling $m = \hat{m} + m_i$, set

$$\widetilde{K}_1 \leftarrow \left[-m\sigma - s\mathbf{a}^\top \mathbf{w}_1\right]_1 \cdot \prod_{i \in Q_0} c_{1,i}, \qquad \widetilde{K}_2 \leftarrow v_2 h^{\widehat{\rho}} H.$$

- $\ \mathsf{H}_4^0 \coloneqq \mathsf{H}_3. \ \mathsf{H}_4^\ell = \mathsf{H}_4^{\ell-1} \ \text{but in the } \ell\text{-th election } H \leftarrow \$ \ \mathbb{G}_1 \ \text{instead of } H \leftarrow g^\theta \, [\sigma]_1.$
- H_5 : As H_4^L but initially set $B, D, E \leftarrow \emptyset$. Upon receiving elect sample $\gamma \leftarrow^{\$} [n]$ and, during the threshold decryption, set $m_i \in [n]$ such that $\gamma = \hat{m} + m_i \mod n$. If $\gamma \notin S$ find $(j, \gamma) \in R$ and store

$$E \leftarrow E \cup \{(eid, j)\}, \ B \leftarrow B \cup \{(eid, K_1, K_2, \widehat{m}, n, R)\}, \ D \leftarrow D \cup \{(eid, \gamma)\}.$$

Finally let P_{ν} return (outcome, eid, 0) if $j \neq \nu$, (outcome, eid, 1) otherwise.

Upon receiving (reveal, *eid*) for honest P_{ν} , if $(eid, \nu) \in E$ find the tuples $(eid, K_1, K_2, \widehat{m}, \widehat{n}, \widehat{R}) \in B$, $(eid, \gamma) \in D$ and compute $m_i \in [\widehat{n}], \delta$ such that $\widehat{m} + m_i = \delta \widehat{n} + \gamma$. Next simulate π_{Dec} and broadcast (claim, *eid*, $\pi_{\mathsf{Dec}}, K_1, K_2, \gamma, \delta$).

Else broadcast (claim, eid, \perp).

- $\mathsf{H}_6^0 \coloneqq \mathsf{H}_5$. H_6^ℓ as $\mathsf{H}_6^{\ell-1}$ but, upon receiving (claim, $\pi, K_1, K_2, \gamma, \delta$) from a corrupted player P_{ν} , honest parties reply with (rejected, eid, ν) if $(eid, \nu) \notin E$.
- $H_7^0 := H_6^L$. H_7^{ℓ} : as $H_7^{\ell-1}$ but in the ℓ -th election, if an honest user wins with a non fully malicious committee, i.e. if $Q \not\subseteq M$ and (j, γ) is chosen from R with $j \notin M$, then set $m_i = -1 \hat{m}$.

Claim 1. $H_{real} \equiv H_0$: From perfect ZK of the arguments used, simulated proofs have the same distribution of correctly generated ones. Hence the two games are identically distributed.

Claim 2. $H_0 \equiv H_0^*$: Observe that in the second game $\widetilde{K}_1 = G_1^x = G_1^{f(-1)}$ and that. Moreover by the soundness of $\pi_{\mathsf{DDH},j}$ the discrete logarithm of $K_{1,j}$ in base G_1 is the same of k_j in base g, i.e. f(j). Hence the decryption shares sent by the honest user P_{ν} are

$$\widetilde{K}_{1,\nu} = G_1^{\lambda_{-1}f(-1)} \cdot \prod_{j \in M} G_1^{\lambda_j f(j)} = G_1^{\lambda_{-1}f(-1) + \sum_{j \in M} \lambda_j f(j)} = G_1^{f(\nu)}$$

where the last equality follows as f has degree t, $|M \cup \{-1\}| = t$ and λ_{-1}, λ_j are the Lagrange coefficients to evaluate in ν . Therefore the decryption share $K_{1,\nu}$ is equal to the one produced in H_0 . As this relation holds in the same way for $K_{2,\nu}$ the claim in proven.

Claim 3. $\mathsf{H}_1^{\ell-1} \equiv \mathsf{H}_1^{\ell}$: Given a PPT distinguisher \mathcal{D} , we provide an algorithm \mathcal{A} that breaks DDH with the same advantage.

Description of $\mathcal{A}(h', u_1, v_1)$: Initially set $\alpha \leftarrow^{\$} \mathbb{F}_q$, $g \leftarrow g_1^{\alpha}$, $h \leftarrow (h')^{\alpha}$. Run $M \leftarrow \mathcal{D}(1^{\lambda}, N)$ which, without loss of generality, we assume to have size |M| = t - 1. Set ecnt $\leftarrow 0$, mpk, msk \leftarrow FE.Setup $(1^{\lambda}, 3)$ and sample $f_j \leftarrow^{\$} \mathbb{F}_q$. Compute $k_j \leftarrow g^{f_j}$ for all $j \in M$ and set $k_{\nu} \leftarrow h^{\lambda_{-1}} \cdot \prod_{j \in M} k_j^{\lambda_j}$ for all $\nu \notin M$. Finally call pp \leftarrow (mpk, $g, h, k_0, \ldots, k_{N-1}$). Upon receiving:

- (setup) from P_{ν} : Reply with (input, pp, f_j).
- (elect, *eid*) from all honest users: update ecnt \leftarrow ecnt + 1, sample $\theta, \sigma \leftarrow^{\$} \mathbb{F}_q$ and Q a random subset of registered users with $|Q| = \kappa$. Call $G \leftarrow g^{\theta}$ and $H \leftarrow h^{\theta}[\sigma]_1$ and let $\mathcal{F}_{\mathsf{CT}}^{\mathsf{ch}}$ return (tossed, *eid*, *G*, *H*, *Q*). If $Q \not\subseteq M$ define $i = \min(Q \setminus M)$ and execute P_i as in the real protocol setting

If ecnt
$$< \ell$$
: $G_{1,i} \leftarrow \tilde{u}_1 G$, $c_{1,i} \leftarrow \tilde{v}_1 H^{m_i} \left[s_i \mathbf{a}^\top \mathbf{w}_1 \right]_1$
If ecnt $= \ell$: $G_{1,i} \leftarrow u_1 G$, $c_{1,i} \leftarrow v_1 H^{m_i} \left[s_i \mathbf{a}^\top \mathbf{w}_1 \right]_1$
If ecnt $> \ell$: $G_{1,i} \leftarrow g^{r_i} G$, $c_{1,i} \leftarrow h^{r_i} H^{m_i} \left[s_i \mathbf{a}^\top \mathbf{w}_1 \right]_1$

with uniformly sampled $m_i, r_i, s_i \leftarrow {}^{\$} \mathbb{F}_q$ and $\widetilde{u}_1, \widetilde{v}_1 \leftarrow {}^{\$} \mathbb{G}_1$.

When \mathcal{D} replies with $(\mathsf{msg}, eid, G_{1,\mu}, G_{2,\mu}, \mathbf{c}_{0,\mu}, c_{1,\mu}, c_{2,\mu}, c_{3,\mu}, \pi_{\mathsf{LR},\mu})$ with accepting proof from $P_{\mu}, \mu \in Q_0$, extract¹³ the witness $s_{\mu}, r_{\mu}, \rho_{\mu}, m_{\mu}$ and let $\hat{s}, \hat{r}, \hat{\rho}, \hat{m}$ be the sum of these variables

 $^{^{13}}$ As ${\cal A}$ is executing ${\cal D}$ as a subroutine the extraction can be performed through rewinding.

for $\mu \in Q_0 \setminus \{i\}$. Compute the decryption shares of honest users setting $\widetilde{K}_2 \leftarrow h^{\widehat{\rho}+\rho_i} \cdot h^{\theta}$ and

$$\begin{split} \text{If ecnt} &< \ell: \ \widetilde{K}_1 \leftarrow \widetilde{v}_1 \cdot h^{\widehat{r}} \cdot h^{\theta(\widehat{m}+m_i)} \\ \text{If ecnt} &= \ell: \ \widetilde{K}_1 \leftarrow v_1 \cdot h^{\widehat{r}} \cdot h^{\theta(\widehat{m}+m_i)} \\ \text{If ecnt} &> \ell: \ \widetilde{K}_1 \leftarrow h^{\widehat{r}+r_i} \cdot h^{\theta(\widehat{m}+m_i)} \end{split}$$

Finally for the remaining phases, i.e. the registration and claim, correctly simulate H_1^{ℓ} .

Proof of Claim: First of all, regardless of the input received by \mathcal{A} , observe that it perfectly simulates the setup calls because, calling x the discrete logarithm of h in base g, by polynomial interpolation there exists a unique $f \in \mathbb{F}_q[x]_{<t}$ such that f(-1) = x and $f(j) = f_j$ for $j \in M$, and in particular f is uniformly distributed over $\mathbb{F}_q[x]_{<t}$. Moreover, by the properties of Lagrange coefficients $k_{\nu} = g^{f(\nu)}$.

Next, when $\operatorname{ecnt} \neq \ell$, the replies to elect requests are distributed as in H_1^ℓ and $\mathsf{H}_1^{\ell-1}$ because $G_{1,i}, c_{1,i}$ are computed as specified by those functionalities and $h^{\theta(\widehat{m}+m_i)} = G^{x(\widehat{m}+m_i)}$ which implies that \widetilde{K}_1 is computed correctly. Similarly as $h^{\theta} = G^x$ also \widetilde{K}_2 is constructed correctly.

Finally, if \mathcal{A} is executed in DDH^0 the elements u_1, v_1 are uniform over \mathbb{G}_1 and so it simulates by definition the functionality H_1^ℓ . Otherwise if \mathcal{A} is executed in DDH^1 , with the above notation $v_1 = u_1^x$. If we let $r_i \in \mathbb{F}_q$ be the discrete logarithm of u_1 in base g, then $u_1 = g^{r_i}, v_1 = h^{r_i}$ implying that for $\mathsf{ecnt} = \ell$

$$G_{1,i} = g^{r_i} G \quad c_{1,i} = h^{r_i} H^{m_i} \left[s_i \mathbf{a}^\top \mathbf{w}_1 \right]_1$$
$$\widetilde{K}_1 = h^{\widehat{r} + r_i} h^{\theta(\widehat{m} + m_i)} = h^{\widehat{r} + r_i} G^{x(\widehat{m} + m_i)}$$

Hence \mathcal{D} distinguishes H_1^{ℓ} from $\mathsf{H}_1^{\ell-1}$ with the same advantage \mathcal{A} has to break DDH in \mathbb{G}_1 , that is negligible.

Claim 4. $\mathsf{H}_2^{\ell-1} \equiv \mathsf{H}_2^{\ell}$: Given a PPT distinguisher \mathcal{D} we provide an algorithm \mathcal{A} that breaks DDH in \mathbb{G}_1 with the same advantage.

Description of $\mathcal{A}(h', u_2, v_2)$: Initially set $\alpha \leftarrow^{\$} \mathbb{F}_q$, $g \leftarrow g_1^{\alpha}$, $h \leftarrow (h')^{\alpha}$. Run $M \leftarrow^{\$} \mathcal{D}(1^{\lambda}, N)$ with |M| = t - 1. Set ecnt $\leftarrow 0$, mpk, msk $\leftarrow^{\$} \mathsf{FE}$.Setup $(1^{\lambda}, 3)$ and sample $f_j \leftarrow^{\$} \mathbb{F}_q$ for $j \in M$. Compute $k_j \leftarrow g^{f_j}$ for $j \in M$ and $k_{\nu} \leftarrow h^{\lambda_{-1}} \prod_{j \in M} k_j^{\lambda_j}$ where λ_j are the Lagrange coefficients associated to $M \cup \{-1\}$ to evaluate in ν . Finally call pp \leftarrow (mpk, $g, h, k_0, \ldots, k_{N-1}$). Upon receiving:

- (setup) from P_i : Reply with (input, pp, f_i).
- (elect, *eid*) from all honest users: update enct \leftarrow ecnt + 1. Sample $\theta, \sigma \leftarrow^{\$} \mathbb{F}_q$ and Q a random subset of the currently registered users with $|Q| = \kappa$. After setting $G \leftarrow g^{\theta}, H \leftarrow h^{\theta} \cdot [\sigma]_1$ return as $\mathcal{F}_{\mathsf{CT}}^{\mathsf{ch}}$ the message (tossed, *eid*, *G*, *H*, *Q*). If $Q \nsubseteq M$ call $i = \min(Q \setminus M)$ and execute P_i as in H_1^L but setting

$$\begin{split} \text{If ecnt} &< \ell: \qquad G_{2,i} \leftarrow \widetilde{u}_2, \qquad c_{2,i} \leftarrow \widetilde{v}_2^{-1} \begin{bmatrix} s_i \mathbf{a}^\top \mathbf{w}_2 \end{bmatrix}_1, \qquad c_{3,i} \leftarrow \widetilde{v}_2^{-n} \begin{bmatrix} s_i \mathbf{a}^\top \mathbf{w}_3 \end{bmatrix}_1 \\ \text{If ecnt} &= \ell: \qquad G_{2,i} \leftarrow u_2, \qquad c_{2,i} \leftarrow v_2^{-1} \begin{bmatrix} s_i \mathbf{a}^\top \mathbf{w}_2 \end{bmatrix}_1, \qquad c_{3,i} \leftarrow v_2^{-n} \begin{bmatrix} s_i \mathbf{a}^\top \mathbf{w}_3 \end{bmatrix}_1 \\ \text{If ecnt} &> \ell: \qquad G_{2,i} \leftarrow g^{\rho_i}, \qquad c_{2,i} \leftarrow h^{-\rho_i} \begin{bmatrix} s_i \mathbf{a}^\top \mathbf{w}_2 \end{bmatrix}_1, \qquad c_{3,i} \leftarrow h^{-n\rho_i} \begin{bmatrix} s_i \mathbf{a}^\top \mathbf{w}_3 \end{bmatrix}_1 \end{split}$$

with uniformly distributed $m_i, \rho_i, s_i \leftarrow \mathbb{F}_q$ and $\widetilde{u}_2, \widetilde{v}_2 \leftarrow \mathbb{G}_1$.

When \mathcal{D} replies with $(\mathsf{msg}, eid, G_{1,\mu}, G_{2,\mu}, c_{0,\mu}, c_{1,\mu}, c_{2,\mu}, c_{3,\mu}, \pi_{\mathsf{LR},\mu})$ with accepting $\pi_{\mathsf{LR},\mu}$ for $\mu \in Q_0$: Extract $s_{\mu}, r_{\mu}, \rho_{\mu}, m_{\mu}$ and let $\hat{s}, \hat{r}, \hat{\rho}, \hat{m}$ be the sum of these variables with $\mu \in Q_0 \setminus \{i\}$. Evaluate the decryption shares of honest users by setting $\widetilde{K}_1 \leftarrow v_1 \cdot h^{\widehat{r}} \cdot h^{\theta(\widehat{m}+m_i)}$ and

If ecnt
$$< \ell$$
: $\widetilde{K}_2 \leftarrow \widetilde{v}_2 \cdot h^{\widehat{\rho}} \cdot h^{\theta}$
If ecnt $= \ell$: $\widetilde{K}_2 \leftarrow v_2 \cdot h^{\widehat{\rho}} \cdot h^{\theta}$
If ecnt $> \ell$: $\widetilde{K}_2 \leftarrow h^{\widehat{\rho}+\rho_i} \cdot h^{\theta}$

Finally in the registration and claim phases correctly simulate H_2^{ℓ} .

Proof of Claim: As in the previous claim, **setup** calls are correctly simulated by polynomial interpolation and the properties of Lagrange coefficients.

Regarding the election phase, when $\operatorname{ecnt} \neq \ell$ the adversary \mathcal{A} perfectly simulates both hybrid functionalities $\mathsf{H}_2^{\ell-1}$ and H_2^{ℓ} , which in this case behave identically. This follows as $G_{1,i}$, $c_{2,i}$ and $c_{3,i}$ are computed as prescribed. Moreover in the threshold decryption step, $G^x = h^{\theta}$ implies that $\widetilde{K}_1 = v_1 h^{\widehat{r}} \cdot G^{x(\widehat{m}+m_i)}$ and

If
$$\operatorname{ecnt} < \ell$$
 : $\widetilde{K}_2 = v_2 \cdot h^{\widehat{\rho}} \cdot G^x$ If $\operatorname{ecnt} > \ell$: $\widetilde{K}_2 = h^{\widehat{\rho} + \rho_i} \cdot G^x$

We now show that when \mathcal{A} is executed in DDH^0 , then it simulates perfectly the functionality H_2^ℓ . This follows as u_2, v_2 are uniformly and independently distributed over \mathbb{G}_1 , so in the ℓ -th election $G_{2,i}$, $c_{2,i}$ and $c_{3,i}$ are generated as in H_1^ℓ and so is $\widetilde{K}_2 = v_2 h^{\widehat{\rho}} G^x$. Conversely when \mathcal{A} is executed in DDH^1 it simulates $\mathsf{H}_2^{\ell-1}$ because $v_2 = u_2^x$. Calling ρ_i the discrete logarithm of u_2 with base g we have that $v_2 = h^{\rho_i}$ with $\rho_i \sim U(\mathbb{F}_q)$ and in particular

$$G_{2,i} = g^{\rho_i}, \qquad c_{2,i} = h^{-(\hat{\rho} + \rho_i)} \left[s_i \mathbf{a}^\top \mathbf{w}_2 \right]_1, \qquad c_{3,i} = h^{-n(\hat{\rho} + \rho_i)} \left[s_i \mathbf{a}^\top \mathbf{w}_3 \right]_1$$

and $\widetilde{K}_2 = h^{\rho_i + \widehat{\rho}} \cdot G^x$, in agreement with the behaviour of $\mathsf{H}_2^{\ell-1}$. Therefore the advantage of \mathcal{D} is equal to the advantage of \mathcal{A} , that is negligible under the assumption that DDH is hard in \mathbb{G}_1 .

Claim 5. $H_2^L \equiv H_3$: To show that the two given functionalities are indistinguishable we will prove them equally distributed. For each election let G, H, Q be the messages sent by \mathcal{F}_{CT}^{ch} with $G = g^{\theta}$, $H = h^{\theta} [\sigma]_1$. If $Q \not\subseteq M$ consider the following change of variables

$$u_1^* = u_1 G^{-m_i}$$
 $u_2^* = u_2$ $v_1^* = v_1 H^{-m_i} \left[-s_i \mathbf{a}^\top \mathbf{w}_1 \right]_1$ $v_2^* = v_2 \left[-\sigma \right]_1$

This defines a map $\varphi : \mathbb{G}_1^4 \to \mathbb{G}_1^4$ such that $\varphi(u_1, u_2, v_1, v_2) = (u_1^*, u_2^*, v_1^*, v_2^*)$. Since this map is a bijection, the image of a uniformly distributed tuple is still uniform over \mathbb{G}_1^3 . Hence, applying the change of variables to H_2^L we have that in each election

$$G_{1,i} = u_1^* \qquad G_{2,i} = u_2^* \qquad c_{1,i} = v_1^*$$
$$c_{2,i} = (v_2^*)^{-1} \begin{bmatrix} -\sigma + s_i \mathbf{a}^\top \mathbf{w}_2 \end{bmatrix}_1 \qquad c_{3,i} = (v_2^*)^{-n} \begin{bmatrix} -n\sigma + s_i \mathbf{a}^\top \mathbf{w}_3 \end{bmatrix}_1$$

Moreover in the threshold decryption phase $\widetilde{K}_2 = v_2^* h^{\widehat{\rho}} G^x[\sigma] = v_2^* h^{\widehat{\rho}} H$. To conclude we need to shown that \widetilde{K}_1 after the variable change is as prescribed in H₃. To this aim observe that

$$\begin{split} \left[m\sigma + s\mathbf{a}^{\top}\mathbf{w}_{1} \right]_{1}^{-1} \cdot \prod_{j \in Q_{0}} c_{1,j} &= \left[m\sigma + s\mathbf{a}^{\top}\mathbf{w}_{1} \right]_{1}^{-1} \cdot h^{\widehat{r}}H^{\widehat{m}} \left[\widehat{s}\mathbf{a}^{\top}\mathbf{w}_{1} \right]_{1} \cdot v_{1}^{*} \\ &= \left[m\sigma + s_{i}\mathbf{a}^{\top}\mathbf{w}_{1} \right]_{1}^{-1} \cdot v_{1}^{*}h^{\widehat{r}}G^{x\widehat{m}} \left[\sigma\widehat{m} \right]_{1} \\ &= \left[m_{i}\sigma + s_{i}\mathbf{a}^{\top}\mathbf{w}_{1} \right]_{1}^{-1} \cdot v_{1}^{*}h^{\widehat{r}}G^{x\widehat{m}} \cdot H^{m_{i}}H^{-m_{i}} \\ &= \left[s_{i}\mathbf{a}^{\top}\mathbf{w}_{1} \right]_{1}^{-1} \cdot v_{1}^{*}h^{\widehat{r}}G^{x(\widehat{m}+m_{i})} \cdot H^{-m_{i}} \\ &= \widetilde{K}_{1}. \end{split}$$

Claim 6. $H_4^{\ell} \equiv H_4^{\ell-1}$: Given a distinguisher \mathcal{D} we provide a PPT algorithm \mathcal{A} that breaks DDH over \mathbb{G}_1 .

Description of $\mathcal{A}(h', u_3, v_3)$: Initially set $\alpha \leftarrow^{\$} \mathbb{F}_q$, $g \leftarrow g_1^{\alpha}$, $h \leftarrow (h')^{\alpha}$. Run $M \leftarrow^{\$} \mathcal{D}(1^{\lambda}, N)$ with |M| = t - 1. Set ecnt $\leftarrow 0$, mpk, msk $\leftarrow^{\$} \mathsf{FE}$.Setup $(1^{\lambda}, 3)$ and sample $f_j \leftarrow^{\$} \mathbb{F}_q$ for $j \in M$. Compute $k_j \leftarrow g^{f_j}$ for $j \in M$ and for $\nu \notin M$ compute k_{ν} interpolating at the exponent as done in the previous claims. Set pp \leftarrow (mpk, $g, h, k_0, \ldots, k_{N-1}$). Upon receiving:

- (setup) from P_j : reply with (input, p, f_j).
- (elect, *eid*) from all honest users: update ecnt \leftarrow ecnt + 1. Sample $\theta, \sigma \leftarrow$ [§] \mathbb{F}_q and compute

If ecnt
$$< \ell$$
: $G \leftarrow^{\$} \mathbb{G}_{1}$, $H \leftarrow^{\$} \mathbb{G}_{1}$
If ecnt $= \ell$: $G \leftarrow u_{3}$, $H \leftarrow v_{3} [\sigma]_{1}$
If ecnt $> \ell$: $G \leftarrow g^{\theta}$, $H \leftarrow h^{\theta} [\sigma]_{1}$

and Q a random subset of registered users with $|Q| = \kappa$. Broadcast the tuple (tossed, *eid*, G, H, Q) and continue the election as in the functionality H₃.

Finally in the registration and claim phase, behave as H_3 .

Proof of Claim: As the previous claim, the replies to setup are correctly handled. Moreover when ecnt $\neq \ell$, G, H are distributed as in H_4^ℓ and $\mathsf{H}_4^{\ell-1}$ by inspection. Finally when \mathcal{A} is executed in DDH^0 , u_3, v_3 are uniformly and independently random, implying that $G, H \sim U(\mathbb{G}_1)$ and independently from σ . Hence \mathcal{A} simulates the functionality H_4^ℓ .

Otherwise, when our algorithm is executed in DDH¹, there exists a $\theta \in \mathbb{F}_q$ such that $u_3 = g^{\theta}$ and $v_3 = h^{\theta}$. By inspection it follows that \mathcal{A} simulated $\mathsf{H}_4^{\ell-1}$ in this case.

In conclusion the advantage of \mathcal{D} in distinguishing the two functionalities equals the negligible advantage of \mathcal{A} in breaking DDH over \mathbb{G}_1 .

Claim 7. $\mathsf{H}_4^L \equiv \mathsf{H}_5$: We begin observing two facts. First of all that R by construction satisfies $(j_0, \gamma), (j_1, \gamma) \in R \Rightarrow j_0 = j_1$. Moreover m_i is correctly distributed as $\gamma \sim U([n])$ implies $m_i = \widehat{m} - \gamma \sim U([n])$.

Next, we show that in H_4^L and H_5 the distribution of (outcome, eid, \cdot) messages from P_{ν} is the same. Since in H_4^L the user P_{ν} looks for a key $\mathsf{sk}_{\gamma,\delta}$ that decrypts the challenge ciphertext and in H_5 it only checks if $(eid, j) \in E$ we now show that this two events are equivalent

 \Leftarrow If $(eid, \nu) \in E$, then during the elections a tuple $(\nu, \gamma) \in R$ was sampled. In particular $m = \gamma \mod n$ and the ciphertext c obtained after the threshold decryption has the following share

$$c = \left([s\mathbf{a}]_1, \ \left[m\sigma + s\mathbf{a}^{\top}\mathbf{w}_1 \right]_1, \ \left[-\sigma + s\mathbf{a}^{\top}\mathbf{w}_2 \right]_1, \ \left[-n\sigma + s\mathbf{a}^{\top}\mathbf{w}_3 \right]_1 \right)$$

i.e. it is an encryption of $(m, -1, -n) = \mathbf{x}_{m,n}$. Since *m* is the sum of m_{μ} with $\mu \in Q_0 \subseteq Q$ and $m_{\mu} \in [n]$ we have that $m \in |Q_0| \cdot [n] \subseteq \kappa[n]$. Therefore calling δ the integer such that $m = \delta n + \gamma$ then $\gamma \in [\kappa]$.

This means that $\mathsf{sk}_{\gamma,\delta} \in \mathcal{K}$ with \mathcal{K} the set of keys received by P_{ν} and that the vector $\mathbf{y}_{\gamma,\delta} = (1, \gamma, \delta)$ is orthogonal to $\mathbf{x}_{m,n}$. Hence, by correctness of the underlying OFE scheme, $\mathsf{FE}.\mathsf{Dec}(c, \mathbf{y}_{\gamma,\delta}, \mathsf{mpk}, \mathsf{sk}_{\gamma,\delta}) \rightarrow 1$.

⇒ If, calling c the challenge ciphertext produced after the threshold decryption, there exists a key $\mathbf{sk}_{\gamma,\delta} \in \mathcal{K}$ such that $\mathsf{FE}.\mathsf{Dec}(c, \mathbf{y}_{\gamma,\delta}, \mathsf{mpk}, \mathsf{sk}_{\gamma,\delta}) \to 1$ then $(\nu, \gamma) \in R$ and c is the encryption of a vector $\mathbf{x}_{m,n}$ orthogonal to $\mathbf{y}_{\gamma,\delta}$. This implies that $m - \gamma - \delta n = 0$ or in other words $m = \gamma \mod n$. Hence a couple $(j, \gamma) \in R$ was sampled during the election, meaning that $(eid, j) \in E$. As $(\nu, \gamma) \in R$ we conclude that $j = \nu$ and $(eid, \nu) \in E$.

The same argument shows that replies to (reveal, eid) are distributed equally in the two functionalities. This proves the claim.

Claim 8. $H_6^{\ell-1} \equiv H_6^{\ell}$: The only way in which the two functionality may be distinguished is if honest players reject a dishonest claim in H_6^{ℓ} but accept it in $H_6^{\ell-1}$. To make this probability negligible we show that an adversary \mathcal{D} capable of producing this effect can be transformed into a PPT algorithm \mathcal{A} trying to break the selective security of our OFE scheme.

Description of \mathcal{A} : Run $M \leftarrow^{\$} \mathcal{D}(1^{\lambda}, N)$ with |M| = t - 1 and initialize ecnt $\leftarrow 0, n \leftarrow 0$. Let p be the maximum number of registration \mathcal{D} performs. Then sample $n^* \leftarrow^{\$} [p + \kappa L], m^* \leftarrow^{\$} [n^*]$ and set γ^*, δ^* such that $\gamma^* \in [n^*]$ and $m^* = \gamma^* + n^* \delta^*$. Define

$$\mathbf{x}_0 = (m^*, -1, -n^*), \quad \mathbf{x}_1 = \kappa n^* \cdot (\kappa \gamma^* + \delta^*, -\kappa, -1)$$

and send these two vectors to the challenger Π . Wait for its reply (mpk, c^*) and compute pp as in H₅. Upon receiving:

- (keygen) from $P_j, j \in M$:

Wait for all elected users to reveal themselves. While $n \in S$ increment n by one. Request from Π the keys $\mathsf{sk}_{n,\delta}$ associated to the vectors $\mathbf{y}_{n,\delta}$, $\delta \in [\kappa]$. Send (key, $\{\mathsf{sk}_{n,\delta}\}$) to P_j and make honest users return (key_requested, j). Finally increase $n \leftarrow n + 1$.

- (elect, *eid*) from honest players: increment ecnt by one. If ecnt $\neq \ell$ execute the election as in H₅, otherwise simulate \mathcal{F}_{CT}^{ch} returning (G, H, Q) with $Q \nsubseteq M$. Sample $\gamma \leftarrow^{\$} [n]$. If $\gamma \in S$ execute

the election setting $m_i = \hat{m} - \gamma$ in the threshold decryption phase. Else, parse $c^* = (\mathbf{c}_0^*, c_1^*, c_2^*, c_3^*)$ and execute P_i with $i = \min(Q \setminus M)$ by setting

$$G_1 \leftarrow u_1, \quad G_2 \leftarrow u_2, \quad \mathbf{c}_0 \leftarrow \mathbf{c}_0^*, \quad c_{1,i} \leftarrow v_1$$
$$c_{2,i} \leftarrow v_2^{-1} c_2^*, \quad c_{3,i} \leftarrow v_2^{-n} c_3^*.$$

Upon receiving $(\mathsf{msg}, G_{1,\mu}, G_{2,\mu}, c_{1,\mu}, c_{2,\mu}, c_{3,\mu}, \pi_{\mathsf{LR},\mu})$ with accepting proof from P_{μ} with $\mu \in Q_0 \subseteq Q$, extract the witness $s_{\mu}, r_{\mu}, \rho_{\mu}, m_{\mu}$ and compute $\hat{s}, \hat{m}, \hat{r}, \hat{m}$ the sum of the respective variable for $\mu \in Q_0$. Return a random bit and abort if the following conditions are not satisfied

$$n = n^*, \qquad m^* - \widehat{m} \in [n]$$

Otherwise execute P_i setting $\widetilde{K}_1 = (c_1^*)^{-1} \left[-\widehat{\mathbf{s}} \mathbf{a}^\top \mathbf{w}_1 \right]_1 \widehat{c}_1$ with \widehat{c} being the product of $c_{1,\mu}$ for $\mu \in Q_0$.

- (claim, $eid, k_1, k_2, \pi, \gamma, \delta$) from P_j : if eid is the ID of the ℓ -th election, check $(j, \gamma) \in R$ and that π is accepting. If $\gamma \neq \gamma^*$ compute

$$\mathbf{c}_0 \leftarrow \prod_{\mu \in Q_0} \mathbf{c}_{0,\mu}, \qquad c_1 \leftarrow K_1^{-1} \cdot \prod_{\mu \in Q_0} c_{1,\mu}, \\ c_2 \leftarrow K_2 \cdot \prod_{\mu \in Q_0} c_{2,\mu}, \qquad c_3 \leftarrow K_2^n \cdot \prod_{\mu \in Q_0} c_{3,\mu}$$

and, calling $\mathbf{y}_{\gamma,\delta} = (y_1, y_2, y_3)$ and $\alpha = (\mathbf{x}_0^\top \mathbf{y}_{\gamma,\delta})^{-1}$,

$$\widetilde{G} \leftarrow \prod_{i \in [3]} \left(c_1^* c_1^{-1} \left[\widehat{s} \mathbf{a}^\top \mathbf{w}_i \right]_1 \right)^{\alpha y_i}$$

Finally, set $\tilde{\mathbf{c}}_0 \leftarrow \mathbf{c}_0^*$, $\tilde{c}_1 \leftarrow c_1^* \cdot \tilde{G}^{-m^*}$, $\tilde{c}_2 \leftarrow c_2^* \cdot \tilde{G}$, $\tilde{c}_3 \leftarrow c_3^* \cdot \tilde{G}^{n^*}$, call $\tilde{c} = (\tilde{\mathbf{c}}_0, \tilde{c}_1, \tilde{c}_2, \tilde{c}_3)$ and request to Π the secret key sk associated to (1, 0, 0). Decrypt $b \leftarrow \mathsf{FE}.\mathsf{Dec}(\tilde{c}, (1, 0, 0), \mathsf{mpk}, \mathsf{sk})$ and return 1 - b

When \mathcal{D} halts return a random bit

Proof of Claim: First of we let \mathcal{E} be the event that all the condition checked in the ℓ -th election by \mathcal{A} are satisfied, i.e. that $n^* = n$ and $m^* - \hat{m} \in [n]$, happens with significant probability. Since \mathcal{D} perform at most p registrations and L elections, we have that after each election S grows by κ elements. Therefore $|S| \leq \kappa L$ which implies $n \leq p + |S| \leq p + \kappa L$ and in particular, as n^* is independent from the value of n during the ℓ -th election, $\Pr[n^* = n] = (p + \kappa L)^{-1}$. Regarding the second condition, observe that \hat{m} is the sum of $\kappa - 1$ elements in [n], therefore its values is contained in $[(\kappa - 1)n]$. It follows that conditioning on $n^* = n$, $\Pr[m^* - \hat{m} \in [n]] =$

$$= \Pr[m^* \in [n] + \widehat{m}] = \frac{|[n] + \widehat{m}|}{|[\kappa n]|} = \frac{n}{\kappa n} = \kappa^{-1}$$

where in the second equality we used the fact that $[n] + \widehat{m} \subseteq [\kappa n]$. We can then conclude that $\Pr[\mathcal{E}] = (p + \kappa L)^{-1} \kappa^{-1}$.

Next we show that conditioning on \mathcal{E} no key request makes Π return \perp . As \mathcal{A} returns a bit after the ℓ -th election is over, all key request occurs before, i.e. they are associated to vectors $\mathbf{y}_{\gamma,\delta} = (1,\gamma,\delta)$ with $\gamma < n^*$.

$$\mathbf{x}_0^\top \mathbf{y}_{\gamma,\delta} = 0 \quad \Rightarrow \quad m^* = \gamma + \delta n^* \quad \Rightarrow \quad (\gamma^* - \gamma) = (\delta - \delta^*) n^*$$

Since both γ and γ^* are smaller than n^* so is the absolute value of their difference, and in particular $\gamma = \gamma^*, \ \delta = \delta^*$ which implies $\mathbf{x}_1^\top \mathbf{y}_{\gamma,\delta}$ by construction. Conversely if $\mathbf{x}_1^\top \mathbf{y}_{\gamma,\delta} = 0$

$$\kappa(\gamma^* - \gamma) = \delta - \delta^* \quad \Rightarrow \quad \gamma^* = \gamma, \quad \delta^* = \delta \quad \Rightarrow \quad \mathbf{x}_0^\top \mathbf{y}_{\gamma, \delta} = 0$$

where we used the fact that δ, δ^* are strictly smaller than κ .

We now define bad the event that \mathcal{D} manage to send (claim, $eid, K_1, K_2, \pi, \gamma, \delta$) with eid the ID of the ℓ -th election, π an accepting proof and $\gamma \neq \gamma^*$. We also call β the random bit chosen by the challenger, i.e. such that c^* is an encryption of \mathbf{x}_{β} with randomness σ and s^* .

We first remark that by inspection, until it halts \mathcal{A} perfectly simulates $\mathsf{H}_6^{\ell-1}$ when $\beta = 0$. It follows that \mathcal{D} behave as in $\mathsf{H}_6^{\ell-1}$ when $\beta = 0$ and, up to negligible probability ε the same happens when $\beta = 1$ as otherwise \mathcal{D} would breaks the selective security of the underlying OFE scheme.

Next we observe than, when **bad** occurs, the simulation soundness of π implies the existence of a key $\mathsf{sk}' = ([\mathbf{d}_0]_2, [d_1]_2)$ which behave as $\mathsf{sk}_{\gamma,\delta}$ and that decrypts c. The first property implies that sk' correctly decrypts and encryption of $\mathbf{0}$, i.e. calling $\mathbf{y}_{\gamma,\delta} = (y_1, y_2, y_3)$ then

$$e\left(\left[\mathbf{a}\right]_{1},\left[\mathbf{d}_{0}\right]_{2}\right) = e\left(\left[\sum_{i=1}^{3} y_{i}\mathbf{a}^{\top}\mathbf{w}_{i}\right]_{1},\left[d_{1}\right]_{2}\right) \quad \Rightarrow \quad \mathbf{a}^{\top}\mathbf{d}_{0} = d_{1}\sum_{i=1}^{3} y_{i}\mathbf{a}^{\top}\mathbf{w}_{i}$$

Using now the fact that $\mathbf{s}\mathbf{k}'$ decrypt c, that is an encryption of \mathbf{x}_{β} with randomness σ and $s = s^* + \hat{s}$, then $e(\mathbf{c}_0, [\mathbf{d}_0]_2) = e(c_1^{y_1} c_2^{y_2} c_3^{y_3}, [d_1]_2)$. Letting $z \in \mathbb{F}_q$ be such that $[z]_1 = c_1^{y_1} c_2^{y_2} c_3^{y_3}$

$$\begin{aligned} s\mathbf{a}^{\top}\mathbf{d}_{0} &= zd_{1} \quad \Rightarrow \quad z = s\sum_{i=1}^{3} y_{i}\mathbf{a}^{\top}\mathbf{w}_{i} \quad \Rightarrow \\ \Rightarrow \quad \widetilde{G} &= \left[\sum_{i=1}^{3} \left(y_{i}x_{\beta,i}\sigma + s^{*}y_{i}\mathbf{a}^{\top}\mathbf{w}_{i}\right) - \left(sy_{i}\mathbf{a}^{\top}\mathbf{w}_{i}\right) + \left(\widehat{s}y_{i}\mathbf{a}^{\top}\mathbf{w}_{i}\right)\right]_{1}^{\alpha} \quad \Rightarrow \\ \Rightarrow \quad \widetilde{G} &= \left[\mathbf{x}_{\beta}^{\top}\mathbf{y}_{\gamma,\delta} \cdot \sigma\right]_{1}^{\alpha} = \left[\alpha \cdot \mathbf{x}_{\beta}^{\top}\mathbf{y}_{\gamma,\delta} \cdot \sigma\right]_{1} \end{aligned}$$

Calling $\alpha_{\beta} = 1 - \alpha \cdot (\mathbf{x}_{\beta}^{\top} \mathbf{y}_{\gamma, \delta})$ we can finally express the components of \tilde{c} as

$$\widetilde{\mathbf{c}} = [s^* \mathbf{a}]_1, \qquad \widetilde{c}_1 = \begin{bmatrix} \alpha_\beta x_{\beta,1} \cdot \sigma + s^* \mathbf{a}^\top \mathbf{w}_1 \end{bmatrix}_1$$
$$\widetilde{c}_2 = \begin{bmatrix} \alpha_\beta x_{\beta,2} \cdot \sigma + s^* \mathbf{a}^\top \mathbf{w}_2 \end{bmatrix}_1, \quad \widetilde{c}_3 = \begin{bmatrix} \alpha_\beta x_{\beta,3} \cdot \sigma + s^* \mathbf{a}^\top \mathbf{w}_3 \end{bmatrix}_1$$

i.e. \tilde{c} is an encryption of the vector $\alpha_{\beta} \mathbf{x}_{\beta}$. Observe that for $\beta = 0$ this factor is 0 as $\alpha_0 = 1 - (\mathbf{x}_0^\top \mathbf{y}_{\gamma,\delta})^{-1} (\mathbf{x}_0^\top \mathbf{y}_{\gamma,\delta}) = 0$. Conversely for $\beta = 1$

$$\mathbf{x}_0^{\top} \mathbf{y}_{\gamma,\delta} = \gamma^* - \gamma + n^* (\delta^* - \delta), \qquad \mathbf{x}_1^{\top} \mathbf{y}_{\gamma,\delta} = n^* \kappa \cdot (\kappa(\gamma^* - \gamma) + \delta^* - \delta)$$

where $\gamma^* \neq \gamma$ implies that both inner products are non-zero and, as they are both polynomially bounded – so no modular reduction occur – and $\mathbf{x}_0^{\top} \mathbf{y}_{\gamma,\delta} < n^* \kappa$. In particular they cannot be equal as $\mathbf{x}_1^{\top} \mathbf{y}_{\gamma,\delta}$ over the integer is a multiple of $n^* \kappa$, so

$$\mathbf{x}_0^{\top} \mathbf{y}_{\gamma,\delta} \neq \mathbf{x}_1^{\top} \mathbf{y}_{\gamma,\delta} \quad \Rightarrow \quad (\mathbf{x}_0^{\top} \mathbf{y}_{\gamma,\delta})^{-1} (\mathbf{x}_1^{\top} \mathbf{y}_{\gamma,\delta}) \neq 1 \quad \Rightarrow \quad \alpha_1 \neq 0$$

We can then conclude that for $\beta = 1$, \tilde{c} contains the encryption of a non-zero multiple of \mathbf{x}_1 , that is not orthogonal to (1, 0, 0). Hence when $\beta = 0$, \tilde{c} can be decrypted with $\mathbf{sk'}$ and \mathcal{A} return 0, while

for $\beta = 1$ the decryption is not successful and \mathcal{A} returns 1. In conclusion

$$\begin{aligned} \mathsf{Adv}(\mathcal{A}) &= \left| \Pr\left[\mathcal{A} \to 0 | \beta = 0\right] - \Pr\left[\mathcal{A} \to 0 | \beta = 1\right] \right| \\ &= \left| \Pr\left[\mathcal{A} \to 0 | \beta = 0, \ \mathcal{E}\right] - \Pr\left[\mathcal{A} \to 0 | \beta = 1, \ \mathcal{E}\right] \right| \cdot \Pr\left[\mathcal{E}\right] \\ &= \left| \frac{1}{2} \Pr\left[\neg\mathsf{bad}\right] + \Pr\left[\mathsf{bad}\right] - \frac{1}{2} \Pr\left[\neg\mathsf{bad}\right] \right| \cdot \Pr\left[E\right] \\ &= \Pr\left[\mathsf{bad}\right] \Pr\left[E\right] \end{aligned}$$

Where the second equality holds as $\Pr[\mathcal{A} \to 0|\beta = 0, \neg \mathcal{E}]$ and $\Pr[\mathcal{A} \to 0|\beta = 0, \mathcal{E}]$ are both equal to 1/2 since in this case \mathcal{A} return a random bit. Hence Pr [bad] is negligible.

Claim 9. $H_7^{\ell-1} \equiv H_7^{\ell}$: Given a distinguisher \mathcal{D} we provide a PPT adversary that breaks the selective security of our OFE scheme whose advantage is a polynomial factor away from the advantage of \mathcal{D} **Description of** \mathcal{A} : Run $M \leftarrow \mathcal{D}$ with |M| = t - 1. Let p be the maximum number of keygen requests \mathcal{D} can make and sample $n^* \leftarrow [p + \kappa L], m^* \leftarrow [\kappa n^*]$. Send to the challenger Π the vectors

$$\mathbf{x}_0 = (m^*, -1, -n^*), \qquad \mathbf{x}_1 = (-1, -1, -n^*)$$

and wait for the reply mpk, c^* . Sample $g, h \leftarrow \ \mathbb{G}_1, f \leftarrow \ \mathbb{F}_q[x]_{\leq t}$ and set $k_{\nu} \leftarrow g^{f(j)}, p \leftarrow$ $(\mathsf{mpk}, g, h, k_0, \dots, k_{N-1})$, ecnt $\leftarrow 0, n \leftarrow 0$ and $E, B, S \leftarrow \emptyset$. Upon receiving from \mathcal{D} :

- (keygen) from $P_j, j \in M$:

Wait for all elected users to reveal themselves. While $n \in S$, increase n by one. Request from Π the secret keys associated to the vectors $\mathbf{y}_{n,\delta}$ for $\delta \in [\kappa]$. If Π replies with \perp abort returning a random bit, otherwise if it replies with $\mathsf{sk}_{n,\delta}$, send (key, $\{\mathsf{sk}_{n,\delta}\}_{\delta \in [\kappa]}$) and let honest users return (key requested, j). Finally increase $n \leftarrow n + 1$.

- (elect, *eid*) from P_i . Set ecnt \leftarrow ecnt + 1. If ecnt $\neq \ell$, execute the election as in $H_7^{\ell-1}$. Else simulate \mathcal{F}_{CT}^{ch} sending G, H, Q and sample $\gamma \leftarrow \ [n].$

If $\gamma \in S$ execute the election as in H_6^L . Else compute γ^*, δ^* with $\gamma^* \in [n^*]$ such that $m^* =$ $\delta^* n^* + \gamma^*$. Next check the following conditions

$$Q \not\subseteq M, \qquad n^* = n, \qquad \exists j^* \notin M : (j^*, \gamma^*) \in R$$

and if any of them is not satisfied, return a random bit and halt. Otherwise set $i = \min(Q \setminus M)$, parse $c^* = (\mathbf{c}_0^*, c_1^*, c_2^*, c_3^*)$ and execute P_i by sampling $u_1, u_2, v_1, v_2 \leftarrow \mathbb{G}_1$ and setting

$$G_{1,i} \leftarrow u_1, \quad G_{2,i} \leftarrow u_2, \quad \mathbf{c}_{0,i} \leftarrow \mathbf{c}_0^*, \quad c_{1,i} \leftarrow v_1$$
$$c_{2,i} \leftarrow v_2^{-1} c_2^* \quad c_{3,i} \leftarrow v_2^{-n} c_3^*.$$

After receiving $(\mathsf{msg}, eid, \mathbf{c}_{0,\mu}, c_{1,\mu}, c_{2,\mu}, c_{3,\mu}, G_{1,\mu}, G_{2,\mu}, \pi_{\mathsf{LR},\mu})$ with accepting proof from P_{μ} for $\mu \in Q_0$, extract $(s_\mu, r_\mu, \rho_\mu, m_\mu)$ the witness of the proof and call $\hat{s}, \hat{r}, \hat{\rho}, \hat{m}$ the sum of the respective variables for $\mu \in Q_0 \setminus \{i\}$.

If $m^* - \hat{m} \notin [n]$ return a random bit and halt, Else perform the threshold decryption setting

$$\widetilde{K}_1 \leftarrow (c_1^*)^{-1} \cdot \left[-\widehat{s} \mathbf{a}^\top \mathbf{w}_1 \right]_1 \cdot \prod_{i \in Q_0} c_{1,i}, \qquad \widetilde{K}_2 \leftarrow v_2 \cdot h^{\widehat{\rho}} \cdot H.$$

Finally update $E \leftarrow E \cup \{(eid, j)\}, B \leftarrow B \cup \{(eid, \widetilde{K}_1, \widetilde{K}_2, \widehat{m}, n, R)\}$ and $D \leftarrow D \cup \{(eid, \gamma)\}.$

When $b \leftarrow \mathcal{D}$, returns b.

Proof of Claim: Let \mathcal{E} denote the event that in the ℓ -th election \mathcal{A} does not abort, i.e. that $Q \not\subseteq M$, $n^* = n$, that there exists $j^* \notin M : (j^*, \gamma^*) \in R$ and $m^* - \widehat{m} \in [n]$. The following properties holds.

 $- \mathcal{E}$ implies that \mathcal{A} does not abort.

Indeed if \mathcal{E} occurs, all the keys requested before the ℓ -th election are associated to vectors of the form $(1, \gamma, \delta)$ for $\gamma < n^*$ and $\delta \in [\kappa]$. Clearly $(1, \gamma, \delta) \cdot (-1, -1, -n^*) \neq 0$ as it is a negative value polynomially bounded, and therefore greater than -q. Hence the challenger returns \perp in this case only if $(1, \gamma, \delta) \cdot (m^*, -1, -n^*) = 0$.

Assume by contradiction that this is the case. Then $m^* = \gamma + \delta n^*$. However by construction $m^* = \gamma^* + \delta^* n^*$ meaning that $\gamma^* - \gamma = (\delta - \delta^*)n^*$. Since both $\gamma, \gamma^* < n^*$ this implies that $\gamma^* - \gamma = 0$.

However, during the key request, (j, γ) is added to R with $j \in M$, while \mathcal{E} implies that there exists $j^* \notin M$ s.t. $(j^*, \gamma^*) \in R$. Hence $j = j^*$ that is a contradiction.

Conversely, keys requested after the ℓ -th election are associated to vectors of the form $(1, \gamma, \delta)$ with $\delta \in [\kappa]$ and $\gamma \geq n^*$.

Assume again by contradiction that one of these vectors is orthogonal to $(m^*, -1, -n^*)$. Then as before this implies that $\gamma = \gamma^* + (\delta^* - \delta)n^*$. Observe that as $\gamma \ge n^* > \gamma^*$, necessarily $\delta^* - \delta > 0$. Moreover $\delta^* - \delta \le \kappa$.

By construction this means that γ was added to S after the ℓ -th election, which is a contradiction as the values in S are skipped while requesting a new key.

- $\Pr[\mathcal{E}] \ge (1-2^{\kappa}) \cdot (p+\kappa L)^{-2} \kappa^{-1}$, i.e. $\Pr[\mathcal{E}]^{-1}$ is polynomially bounded.

First by construction $\Pr[Q] \notin M \ge (1-2^{\kappa})$. Next, as \mathcal{D} performs at most p registrations, of either honest or dishonest users, and after each election the size of S grows at most of κ . It follows that the ℓ -th election n can at most be $p + \kappa \ell \le p + \kappa \ell$. Since \mathcal{D} up to that point had no information on n^* , $\Pr[n = n^*] = (p + \kappa L)^{-1}$.

Next conditioning on $n^* = n$ we observe that \mathcal{D} has no information on m^* either, as the only component of the ciphertext that depends on m^* is c_1^* . Hence \widehat{m}, m^* are independent and

$$\Pr[m^* - \hat{m} \in [n]] = \frac{|[n] + \hat{m}|}{|[\kappa n]|} = \frac{n}{\kappa n} = \kappa^{-1}.$$

Where we used the fact that $\widehat{m} \leq (\kappa - 1)n$ implies $[n] + \widehat{m} \subseteq [\kappa n]$.

Finally conditioning on $m^* \in [n] + \hat{m}$ we have that m^* is uniformly distributed over this set, and in particular $\gamma^* = m^* \mod n$ is uniform over [n]. Since there is at least one honest user registered during the ℓ -th election¹⁴, there exist $(j_0^*, \gamma_0^*) \in R$ such that $j_0^* \notin M$. Therefore

$$\Pr\left[\exists j^* \notin M : (j^*, \gamma^*) \in R\right] \ge \Pr\left[\gamma^* = \gamma_0^*\right] = \frac{1}{n} \ge \frac{1}{(p+\kappa L)}.$$

This concludes the proof of the second property.

¹⁴ In fact we assume that half of the registered player are honest at any time, however this does not imply that with probability 1/2 the value γ^* is associated to an honest user, as there are values in [n] that are not associated to any player.

Next we define the event HW^{ℓ} stating that in $\mathsf{H}_7^{\ell-1}$ (or in H_7^{ℓ}) during the ℓ -th election, calling *eid* the id of this election, $Q \nsubseteq M$ and that $(eid, j) \in E$ with $j \notin M$.

With this notation we argue that when c^* is the encryption of \mathbf{x}_0 , if the event \mathcal{E} occurs, \mathcal{A} simulates $\mathsf{H}_7^{\ell-1}$ conditioned to HW_0^{ℓ} . Indeed, \mathcal{E} implies that the key request are handled correctly and during the ℓ -th election, calling

$$\mathbf{c}_{0}^{*} = [s_{i}\mathbf{a}]_{1}, \quad c_{1}^{*} = \left[\sigma m^{*} + s_{i}\mathbf{a}^{\top}\mathbf{w}_{1}\right]_{1},$$
$$c_{2}^{*} = \left[-\sigma + s_{i}\mathbf{a}^{\top}\mathbf{w}_{2}\right]_{1}, \quad c_{3}^{*} = \left[-n^{*}\sigma + s_{i}\mathbf{a}^{\top}\mathbf{w}_{3}\right]_{1}$$

for uniformly sampled $\sigma, s_i \sim U(\mathbb{F}_q)$, the message sent by P_i are

$$\mathbf{c}_{0,i} = [s_i \mathbf{a}]_1, \quad c_{2,i} = v_2^{-1} \left[-\sigma + s_i \mathbf{a}^\top \mathbf{w}_2 \right]_1, \quad c_{3,i} = v_3^{-n} \left[-n\sigma + s_i \mathbf{a}^\top \mathbf{w}_3 \right]_1$$

Finally in the threshold decryption we have that, calling \hat{c}_1 the products of all $c_{1,j}$ for $j \in Q_0 \setminus \{i\}$,

$$\widetilde{K}_1 = \left[-\sigma m^* - s_i \mathbf{a}^\top \mathbf{w}_1\right]_1 \cdot \left[-\widehat{s} \mathbf{a}^\top \mathbf{w}_1\right]_1 \cdot \widehat{c}_1 = \left[-\sigma m^* - s \mathbf{a}^\top \mathbf{w}_1\right]_1 \cdot \widehat{c}_1$$

Where, calling $m_i = m^* - \widehat{m} \in [n], \ m^* = m_i + \widehat{m}$.

Conversely if c^* is the encryption of \mathbf{x}_1 and \mathcal{E} occurs, \mathcal{A} simulates H_7^ℓ conditioned to HW_1^ℓ . Observe that in this case the only element that is distributed differently is c_1^* , that, with the same notation used above, is $[-\sigma + s_i \mathbf{a}^\top \mathbf{w}_1]_1$. For this reason we only need to check the correctness of \widetilde{K}_1 . Indeed

$$\widetilde{K}_1 = \left[\boldsymbol{\sigma} - s_i \mathbf{a}^\top \mathbf{w}_1 \right]_1 \cdot \left[-\widehat{s} \mathbf{a}^\top \mathbf{w}_1 \right]_1 \cdot \widehat{c}_1 = \left[\boldsymbol{\sigma} - s \mathbf{a}^\top \mathbf{w}_1 \right]_1 \cdot \widehat{c}_1.$$

We can therefore conclude that the claim is true because, calling \mathbf{x}_{β} the vector chosen by the challenger Π

$$\begin{aligned} \mathsf{Adv}(\mathcal{D}) &= \left| \Pr\left[\mathcal{D} \to 0 \mid \mathsf{H}_{7}^{\ell-1} \right] - \Pr\left[\mathcal{D} \to 0 \mid \mathsf{H}_{7}^{\ell} \right] \right| \\ &\leq \left| \Pr\left[\mathcal{D} \to 0 \mid \mathsf{H}_{7}^{\ell-1}, \ \neg \mathsf{HW}^{\ell} \right] - \Pr\left[\mathcal{D} \to 0 \mid \mathsf{H}_{7}^{\ell-1}, \ \neg \mathsf{HW}^{\ell} \right] \right| \cdot \Pr\left[\neg \mathsf{HW}^{\ell} \right] \\ &+ \left| \Pr\left[\mathcal{D} \to 0 \mid \mathsf{H}_{7}^{\ell-1}, \ \mathsf{HW}^{\ell} \right] - \Pr\left[\mathcal{D} \to 0 \mid \mathsf{H}_{7}^{\ell-1}, \ \mathsf{HW}^{\ell} \right] \right| \cdot \Pr\left[\mathsf{HW}^{\ell} \right] \\ &\leq \left| \Pr\left[\mathcal{D} \to 0 \mid \mathsf{H}_{7}^{\ell-1}, \ \mathsf{HW}^{\ell} \right] - \Pr\left[\mathcal{D} \to 0 \mid \mathsf{H}_{7}^{\ell-1}, \ \mathsf{HW}^{\ell} \right] \right| \\ &= \left| \Pr\left[\mathcal{A} \to 0 \mid \beta = 0, \ \mathcal{E} \right] - \Pr\left[\mathcal{A} \to 0 \mid \beta = 1, \ \mathcal{E} \right] \end{aligned}$$

where in the second inequality we used the fact that $\Pr\left[\mathsf{HW}^{\ell}\right] \leq 1$ and that the functionalities H_{7}^{ℓ} and $\mathsf{H}_{7}^{\ell-1}$ are identical if $\neg \mathsf{HW}^{\ell}$. This implies

$$\begin{aligned} \mathsf{Adv}(\mathcal{A}) &= |\Pr\left[\mathcal{A} \to 0 | \beta = 0\right] - \Pr\left[\mathcal{A} \to 0 | \beta = 1\right]| \\ &\geq |\Pr\left[\mathcal{A} \to 0 | \beta = 0, \ \mathcal{E}\right] - \Pr\left[\mathcal{A} \to 0 | \beta = 1, \ \mathcal{E}\right]| \cdot \Pr\left[\mathcal{E}\right] \\ &\geq \mathsf{Adv}(\mathcal{D}) \cdot \Pr\left[\mathcal{E}\right]. \end{aligned}$$

Claim 10. $H_7^L \equiv \mathcal{F}_{SSLE}^{\kappa,\eta} \circ \mathcal{S}$: Follows by inspection.

D.4 Proof of Theorem 3

Proof of Theorem 3. As in the previous proofs we need to show the existence of a simulator such that $S \circ \mathcal{F}_{\mathsf{SK}}$ is indistinguishable from the real protocol for any \mathcal{Z} satisfying the hypothesis.

Description of S:

Setup Phase. Initially wait for $M \leftarrow^{\$} Z$. For each uncorrupted user P_j generate $(\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow^{\$} \mathsf{AE.Setup}(1^{\lambda})$ and broadcast $(\mathsf{user_key}, \mathsf{pk}_j)$. Upon receiving $(\mathsf{setup_leak}, \widetilde{\mathsf{mpk}}, g, \widetilde{h}, (\widetilde{k}_j)_{j=0}^{N-1}, (\widetilde{s}_j)_{j\in M})$ from $\mathcal{F}_{\mathsf{SK}}$, sample a set $Q \subseteq [N]$ of size λ and abort if $Q \subseteq M$.

Parse $\mathsf{mpk} = (\mathbf{z}_0, \widetilde{z}_1, \widetilde{z}_2, \widetilde{z}_3)$, broadcast (tossed, Q, \mathbf{z}_0, g) and simulate honestly all players but P_i . For P_i set

$$\begin{split} z_{\alpha,i} \leftarrow \widetilde{z}_{\alpha} \quad \forall \alpha \in [3], \qquad h_i \leftarrow \widetilde{h}, \qquad k_{j,i} \leftarrow \widetilde{k}_{j,i} \quad \forall j \in [N] \setminus M \\ c_{j,i} \leftarrow \begin{cases} \mathsf{AE}.\mathsf{Enc}(\widetilde{s}_j,\mathsf{pk}_j) & \text{If } j \in M \\ \mathsf{AE}.\mathsf{Enc}(0,\mathsf{pk}_j) & \text{If } j \notin M \end{cases} \end{split}$$

and broadcast $(\mathsf{msg}, h_i, k_{j,i}, c_{j,i}, \pi_{j,i})_{j=0}^{N-1}$ with simulated proof $\pi_{j,i}$ and $(\mathsf{proof}, i, z_{\alpha,i})_{\alpha=1}^3$ as $\mathcal{F}_{\mathsf{zk}}^{\mathcal{R}_{\mathsf{Lin}}}$. Then wait for $(\mathsf{msg}, h_\mu, k_{j,\mu}, c_{j,\mu}, \pi_{j,\mu})$ and $(\mathsf{prove}, \mu, z_{\alpha,\mu}, \mathbf{w}_{\alpha,\mu})$ from $P_\mu, \mu \in Q_0 \cap M$. Run all users correctly, set

$$\begin{split} \mathbf{w}_{\alpha}^{*} &\leftarrow \sum_{\mu \in Q_{0} \setminus \{i\}} \mathbf{w}_{\alpha,\mu}, \qquad s_{j,\mu} \leftarrow \mathsf{AE}.\mathsf{Dec}(c_{i,\mu},\mathsf{sk}_{j}) \\ f_{\mu} &\in \mathbb{F}_{q}[x]_{< t} \ : \ f_{\mu}(j) = s_{j,\mu}, \qquad f^{*} = \sum_{\mu \in Q_{0} \setminus \{i\}} f_{\mu} \end{split}$$

and send (setup_infl, $\mathbf{w}^*_{\alpha}, f^*$) to $\mathcal{F}_{\mathsf{SK}}$.

Key Generation Phase. Upon receiving:

- A request from $\mathcal{F}_{\mathsf{SK}}$ to send (key_requested, j), $j \notin M$: Send (key_request) as P_j , sample $d_{\delta} \leftarrow^{\$} \mathbb{G}_1$ and broadcast (tossed, $rid|j, d_{\delta}$) as $\mathcal{F}_{\mathsf{CT}}^{\mathsf{sk}}$. Let $\mathcal{F}_{\mathsf{SK}}$ send his message and run P_i setting $c_i \leftarrow \mathsf{AE}.\mathsf{Enc}(\mathbf{0},\mathsf{sk}_j)$ and simulating π_i .
- (key_request) from P_j : Send (keygen) to $\mathcal{F}_{\mathsf{SK}}$ as P_j . Upon receiving (key, $\mathsf{sk}_{n,\delta}$) from $\mathcal{F}_{\mathsf{SK}}$ parse $\mathsf{sk}_{n,\delta} = (\mathbf{d}_{\delta}, d_{\delta})$ and broadcast (tossed, $rid|j, d_{\delta}$). Execute P_i by setting

$$\mathbf{d}_{n,\delta}^{(i)} \leftarrow \mathbf{d}_{\delta} \cdot \left[-\sum_{\mu \in Q_n \setminus \{i\}} \mathbf{w}_{1,\mu} + n \mathbf{w}_{2,\mu} + \delta \mathbf{w}_{3,\mu} \right]_{d_{\delta}}$$

- (key_complain, $\mu, n, \delta, \text{sk}, \pi$) from P_j . If π is accepted, and sk fails the check defined on line 28 Protocol 6, send (infl, $\mathbf{w}_{1,\mu}, \mathbf{w}_{2,\mu}, \mathbf{w}_{3,\mu}$) to \mathcal{F}_{SK} .

Hybrid Games: Next, given a PPT environment \mathcal{Z} that performs at most L registrations, we define a sequence of hybrid functionalities:

- H_{real} : The real protocol.

- H₀: If $Q \subseteq M$ in the setup, aborts. All proof produced by honest users are simulated.
- $H_{1,0} \coloneqq H_0$. $H_{1,j}$ as $H_{1,j-1}$ but each time a honest user P_k would send an encrypted message to another honest user P_j , it encrypts **0**. In this case, instead of decrypting the received ciphertext, P_j uses the value P_k would have encrypted.
- H_2 . Initially store accepted witnesses $\mathbf{w}_{\alpha,\mu}$ sent to $\mathcal{F}_{\mathsf{zk}}$ by users in Q and set \mathbf{w}_{α} the sum for $\mu \in Q$ of $\mathbf{w}_{\alpha,\mu}$.

When some user sends (key_complain, μ, \ldots) with accepting proof and satisfying the check on line 28, Protocol 6, update $\mathbf{w}_{\alpha} \leftarrow \mathbf{w}_{\alpha} - \mathbf{w}_{\alpha,\mu}$.

When a honest user P_i request a key, after the key generation phase, it sets

$$\mathsf{sk}_{n,\delta} \;=\; \left[(\mathbf{w}_1 + n\mathbf{w}_2 + \delta\mathbf{w}_3) \cdot r
ight]_1, \; [r]_1$$

and returns $(\text{key}, (\text{sk}_{n,\delta})_{\delta=0}^{\kappa-1})$.

- H_3 : As H_2 but after every key request from a dishonest user P_j , calling $i = \min(Q \setminus M)$, P_i is executed setting $\mathbf{d}_{n,\delta} \leftarrow [\mathbf{w}_1 + n\mathbf{w}_2 + \delta\mathbf{w}_3]_{d_s}$ and

$$\mathbf{d}_{n,\delta}^{(i)} \leftarrow \mathbf{d}_{n,\delta} \cdot \left[-\sum\nolimits_{\mu \in Q \setminus \{i\}} \mathbf{w}_{1,\mu} + n \mathbf{w}_{2,\mu} + \delta \mathbf{w}_{3,\mu} \right]_{d_{\delta}}$$

- H₄: as H₃ but in the setup phase, instead of storing k_j as the product of $k_{j,\mu}$ for $\mu \in Q$, find a polynomial $f \in \mathbb{F}_q[x]_{\leq t}$ such that $f(j) = s_j$ for all $j \in [N] \setminus M$. If no such f exists, abort. Store $h \leftarrow g^{f(-1)}, k_j \leftarrow g^{f(j)}$.

Claim 1. $\mathsf{H}_{\mathsf{real}} \equiv \mathsf{H}_0$: First of all observe that by constructions $|Q| = \lambda$ and that by our assumptions on \mathcal{Z} , |M| < N/2. Hence $\Pr[Q \subseteq M] \le 2^{-\lambda}$ that is negligible. Moreover by the perfect HVZK of the arguments used, simulated proof follows the same distribution of honestly generated ones. Hence the two functionality produce statistically close views.

Claim 2. $H_{1,j-1} \equiv H_{1,j}$: Given a distinguisher \mathcal{D} we sketch an adversary \mathcal{A} that breaks the IND – CPA security of the underlying encryption scheme.

To simplify the reduction we assume that \mathcal{A} has access to an encryption oracle $\mathcal{O}_{\mathsf{Enc}}(m_0, m_1)$ that in the experiment $b \in \{0, 1\}$ returns an encryption of m_b . Through a sequence of standard hybrid games this notion can be shown equivalent to the standard IND – CPA security game.

Description of \mathcal{A} : Run $M \leftarrow \mathcal{F}(1^{\lambda}, N)$. If $j \in M$ return a random bit, otherwise wait for the challenger Π to send pk_j .

Generate $\pi_{\nu}, \mathsf{sk}_{\nu} \leftarrow^{\$} \mathsf{AE.Setup}(1^{\lambda})$ for all $\nu \notin M \cup \{j\}$ and broadcast (user_key, \mathsf{pk}_{\nu}) for $\nu \notin M$.

Setup Phase: Simulate $\mathcal{F}_{\mathsf{CT}}^{\mathsf{stp}}$ returning (tossed, Q, \mathbf{z}_0, g) and for all $i \in Q \setminus M$, execute P_i honestly by setting $c_{j,i} \leftarrow \mathcal{O}_{\mathsf{Enc}}(f_i(j), 0)$.

When \mathcal{D} sends $h_{\mu}, k_{j,\mu}, c_{j,\mu}, \pi_{j,\mu}$ as P_{μ} , if $\pi_{j,\mu}$ is accepted, extract $s_{j,\mu}$ from $\pi_{j,\mu}$. Execute P_j correctly.

Key Distribution Phase: When \mathcal{D} sends (keygen) to P_j , broadcast (key_request) and return (key_requested, j) from any honest user. Sample $d_{\delta} \leftarrow^{\$} \mathbb{G}_2$ for $\delta \in [\kappa]$ and send (tossed, $rid|j, (d_{\delta})_{\delta=0}^{\kappa-1}$).

For all $i \in Q \setminus M$, run P_i by computing honestly $\mathbf{d}_n^{(i)}$ and setting $c_i \leftarrow \mathcal{O}_{\mathsf{Enc}}(\mathbf{d}_n^{(i)}, \mathbf{0})$. When P_{μ} , for $\mu \in Q \cap M$, sends (key_partial, c_{μ}, π_{μ}, j, n), if π_{μ} is accepting run the extractor to get $(\mathbf{d}_{n,\delta}^{(\mu)})_{\delta=0}^{\kappa-1}$ and the randomness used for the encryption. Keep running P_j as prescribed in H_0 .

Output: When $b \leftarrow {}^{\$} \mathcal{D}$, return b.

Proof of Claim: First of all we observe that by weak simulation extractability of the NIZK argument for \mathcal{R}_{Enc} , since all proofs have to be different from previous one¹⁵ the extractor given (c, pk_j) always return, up to negligible probability, m, r such that $c = \mathsf{AE}.\mathsf{Enc}(m, \mathsf{pk}_j; r)$. By perfect correctness of the underlying encryption scheme then, $\mathsf{AE}.\mathsf{Dec}(c, \mathsf{sk}_j) \to m$, which implies that the decryption step performed by P_j is correctly simulated up to negligible probability.

Hence it follows that when \mathcal{A} is executed in the experiment 0, i.e. when $\mathcal{O}_{\mathsf{Enc}}(\cdot, \cdot)$ returns an encryption of its first argument, \mathcal{A} simulated $\mathsf{H}_{1,j-1}$. Conversely if \mathcal{A} is run in experiment 1, then honest users always encrypt the zero vector and in particular it simulates the functionality $\mathsf{H}_{1,j}$.

Claim 3. Let $\mathbf{w}_{\alpha,\mu}$ be the values initially sent to $\mathcal{F}_{\mathsf{zk}}^{\mathcal{R}_{\mathsf{Lin}}}$, and \mathbf{w}_{α} their respective sum for $\mu \in Q$. Then, calling $\mathsf{bad}_{\mathcal{Z}}$ the event that a PPT machine \mathcal{Z} interacting with $H_{2,0}$ makes it return from P_j an uncorrupted user the message

$$(\mathsf{key}, (\mathsf{sk}_{n,\delta})_{\delta=0}^{\kappa-1}) \quad : \quad \exists \delta : \, \mathsf{sk}_{n,\delta} \neq ([\mathbf{w}_1 + n\mathbf{w}_2 + \delta\mathbf{w}_3]_d, d),$$

the probability that bad occurs is negligible.

We prove the claim by providing, for any \mathcal{Z} , an algorithm \mathcal{A} that breaks DDH over \mathbb{G}_2 with advantage almost $\Pr[\mathsf{bad}_{\mathcal{Z}}]$.

Description of $\mathcal{A}(\tilde{u}_2, \tilde{v}_1, \tilde{v}_2)$: Sample $\rho \leftarrow {}^{\$} \mathbb{F}_q$ and set $\mathbf{u} \leftarrow (g_1^{\rho}, \tilde{u}_2^{\rho}), \mathbf{v} \leftarrow (\tilde{v}_1, \tilde{v}_2)$. Run $M \leftarrow {}^{\$} \mathcal{Z}(1^{\lambda})$.

Simulate correctly the initial encryption key generation. Sample $Q \subseteq [N]$, aborting if $Q \subseteq M$, $g \leftarrow^{\$} \mathbb{G}_2$ and $\mathbf{z}_0 \leftarrow \mathbf{u}$. Broadcast (tossed, g, \mathbf{z}_0, Q) and execute according to $\mathsf{H}_{1,N-1}$ honest users, storing the values $\mathbf{w}_{\alpha,\mu}$ for $\mu \in Q$ sent to $\mathcal{F}_{\mathsf{zk}}^{\mathcal{R}_{\mathsf{Lin}}}$.

Each time honest users remove an element from Q, update $\mathbf{w}_{\alpha} \leftarrow \sum_{\mu \in Q} \mathbf{w}_{\alpha,\mu}$.

When an honest user P_j would return $(\text{key}, (\text{sk}_{n,\delta})_{\delta=0}^{\kappa-1})$, parse $\text{sk}_{n,\delta} = (\mathbf{d}, d)$ and set

$$\mathbf{t} \leftarrow \mathbf{d} \cdot [\mathbf{w}_1 + n\mathbf{w}_2 + \delta \mathbf{w}_3]_d$$

If $\mathbf{t} \neq (1, 1)$, set $b \leftarrow e(\mathbf{v}, \mathbf{t}) \stackrel{?}{=} 1$. Return b.

When \mathcal{Z} halts, return a random bit.

Proof of Claim: First of all observe that \mathcal{A} perfectly simulates $\mathsf{H}_{1,N-1}$ regardless of its input distribution because $\widetilde{u}_2 \sim U(\mathbb{G}_1)$ implies that $(g_1^{\rho}, \widetilde{u}_2^{\rho}) \sim U(\mathbb{G}_1^2)$.

Next, \mathcal{A} returns a random bit if and only if $\mathsf{bad}_{\mathcal{Z}}$ does not occur. Indeed, the first event occurs if and only if during the execution of \mathcal{Z} a simulated honest user would return a key $\mathsf{sk}_{n,\delta} = (\mathbf{d}, d)$ such that

 $(1,1) \neq \mathbf{t} = \mathbf{d} \cdot [\mathbf{w}_1 + n\mathbf{w}_2 + \delta\mathbf{w}_3]_d \iff \mathbf{d} \neq [\mathbf{w}_1 + n\mathbf{w}_2 + \delta\mathbf{w}_3]_d.$

¹⁵ we recall that this is obtained by adding a unique session ID, and possibly the identity of the prover, to the proven statement – which applying Fiat-Shamir means that the hash function has to be properly salted.

Finally, assuming that $\mathsf{bad}_{\mathcal{Z}}$ occurs, for any $\mu \in Q$, the honest user that returned the faulty key $\mathsf{sk}_{n,\delta}$ did not complain about P_{μ} 's contribute $\mathbf{d}_{n,\delta}^{(\mu)}$. This means that the check in line 28 pass, i.e.

$$e(\mathbf{z}_{0}, \mathbf{d}_{n,\delta}^{(\mu)}) = e(z_{1,\mu} \cdot z_{2,\mu}^{n} \cdot z_{3,\mu}^{\delta}, d)$$
$$= e(\mathbf{z}_{0}^{\mathbf{w}_{1,\mu} + n\mathbf{w}_{2,\mu} + \delta\mathbf{w}_{3,\mu}}, d)$$
$$= e\left(\mathbf{z}_{0}, [\mathbf{w}_{1,\mu} + n\mathbf{w}_{2,\mu} + \delta\mathbf{w}_{3,\mu}]_{d}\right)$$

where the second equality follows by the definition of $\mathcal{F}_{zk}^{\mathcal{R}_{\text{Lin}}}$. Taking on both sides the product for $\mu \in Q$ and using the bilinearty of e we obtain that

$$e(\mathbf{z}_0, \mathbf{d} \cdot [\mathbf{w}_1 + n\mathbf{w}_2 + \delta \mathbf{w}_3]_d) = 1 \quad \Rightarrow \quad e(\mathbf{u}, \mathbf{t}) = 1$$

where we used the fact that $\mathbf{u} = \mathbf{z}_0$. To conclude observe that in $\mathsf{DDH}^1 \mathbf{v}$ would be proportional to \mathbf{u} , an in particular $e(\mathbf{v}, \mathbf{t}) = 1$. Conversely in DDH^0 , \mathbf{v} is uniformly random and is independent from any coin tossed by \mathcal{A} while executing \mathcal{Z} . In particular \mathbf{v}, \mathbf{t} are independent and $\Pr[e(\mathbf{v}, \mathbf{t}) = 1] = q^{-1}$. Therefore

$$\begin{split} \mathcal{A}(\mathcal{A}) &= \left| \Pr\left[\mathcal{A} \to 1 | \mathsf{DDH}^0\right] - \Pr\left[\mathcal{A} \to 1 | \mathsf{DDH}^1\right] \right| \\ &= \left| \Pr\left[\mathcal{A} \to 1 | \mathsf{bad}_{\mathcal{Z}}, \mathsf{DDH}^0\right] - \Pr\left[\mathcal{A} \to 1 | \mathsf{bad}_{\mathcal{Z}}, \mathsf{DDH}^1\right] \right| \Pr\left[\mathsf{bad}_{\mathcal{Z}}\right] \\ &= \frac{q-1}{q} \cdot \Pr\left[\mathsf{bad}_{\mathcal{Z}}\right] \end{split}$$

where in the second equation we used the independence of $\mathsf{bad}_{\mathcal{Z}}$ from the experiment in which \mathcal{A} is executed in and the fact that when $\mathsf{bad}_{\mathcal{Z}}$ does not happen \mathcal{A} makes a random guess, which gives no advantage.

Claim 4. $H_2 \equiv H_{1,N-1}$: Follows immediately from the previous claim as the only case in which the two world differ is when, for a given adversary \mathcal{Z} , $\mathsf{bad}_{\mathcal{Z}}$ occurs.

Claim 5. $H_3 \equiv H_2$: Follows observing that by construction $\mathbf{w}_{\alpha} = \sum_{\mu \in Q} \mathbf{w}_{\alpha,\mu}$. Hence $\mathbf{w}_{\alpha,i} = \mathbf{w}_{\alpha} - \sum_{\mu \in Q \setminus \{i\}} \mathbf{w}_{\alpha,\mu}$ and in particular $\mathbf{d}_{n,\delta}^{(i)} = [\mathbf{w}_{1,i} + n\mathbf{w}_{2,i} + \delta\mathbf{w}_{3,i}]_{d_{\delta}}$ in both functionalities.

Claim 6. $H_3 \equiv H_4$: We will show that the view generated interacting with this two functionalities is statistically close. To this aim let $\tilde{s}_{j,\mu}, \tilde{s}_{-1,\mu} \in \mathbb{F}_q$ be the discrete logarithm of $k_{j,\mu}$ and h respectively in base g. Calling $\tilde{\mathbf{s}}_{\mu} = (\tilde{s}_{j,\mu})_{j=-1}^{N-1}$ we claim that, if $\tilde{\mathbf{s}}_{\mu} \notin \mathsf{RS}_{\mathbb{F},N+1,t}$ then up to negligible probability all honest users will remove μ from Q.

To see this observe that when $\tilde{\mathbf{s}}$ does not belong in the prescribed Reed Solomon code, it is not orthogonal to all elements of the dual. In particular the subset of vectors in $\mathsf{RS}_{\mathbb{F},N+1,t}^{\perp}$ orthogonal to $\tilde{\mathbf{s}}$ is a proper subspace of co-dimension 1. Hence, given a random vector $\mathbf{v} \leftarrow^{\$} \mathsf{RS}_{\mathbb{F},N+1,t}^{\perp}$

$$\Pr\left[\mathbf{k}_{\mu}^{\mathbf{v}}=1\right] = \Pr\left[g^{\widetilde{\mathbf{s}}_{\mu}^{\top}\mathbf{v}}=1\right] = \Pr\left[\widetilde{\mathbf{s}}_{\mu}^{\top}\mathbf{v}=0\right] = q^{-1}.$$

Using a union bound, the probability that all honest users will remove μ from Q is smaller than Nq^{-1} , and in particular they will remove from Q all μ such that $\tilde{s}_{\mu} \notin \mathsf{RS}_{\mathbb{F},N+1,t}$ with probability smaller than λNq^{-1} .

Conditioning on the latter event for all the remaining $\mu \in Q$ there exists a polynomial $f_{\mu} \in \mathbb{F}_q[x]_{<t}$ such that $h_{\mu} = g^{f_{\mu}(-1)}$ and $k_{j,\mu} = g^{f_{\mu}(j)}$. After the complain step we also have that the element $s_{j,\mu}$ received by honest users are such that $k_{j,\mu} = g^{s_{j,\mu}}$. Hence

$$s_{j,\mu} = f_{\mu}(j) \quad \Rightarrow \quad s_j = \sum_{\mu \in Q} s_{j,\mu} = \sum_{\mu \in Q} f_{\mu}(j)$$

and in particular, since $t = \lfloor N/2 \rfloor$, interpolating s_j for $j \in [N] \setminus M$ yields $f = \sum_{\mu \in Q} f_{\mu}$. This means that honest parties in H_4 will store

$$h = g^{f(-1)} = g^{\sum_{\mu \in Q} f_{\mu}(-1)} = \prod_{\mu \in Q} g^{f_{\mu}(-1)} = \prod_{\mu \in Q} h_{\mu}$$
$$k_{j} = g^{f(j)} = g^{\sum_{\mu \in Q} f_{\mu}(j)} = \prod_{\mu \in Q} g^{f_{\mu}(j)} = \prod_{\mu \in Q} k_{j,\mu}.$$

Claim 7. $H_4 \equiv \mathcal{F}_{SK} \circ \mathcal{S}$: follows by inspection.