Universally Verifiable MPC and IRV Ballot Counting

Chris Culnane[1], Olivier Pereira[1,2], Kim Ramchen[1,3], and Vanessa Teague[1]

[1] Department of Computing and Information Systems
the University of Melbourne
[2] ICTEAM – UCLouvain
B-1348 Louvain-la-Neuve, Belgium
[3] Faculty of Information Technology – Monash University
Clayton, Australia

Abstract. We present a very simple universally verifiable MPC protocol. The first component is a threshold somewhat homomorphic cryptosystem that permits an arbitrary number of additions (in the source group), followed by a single multiplication, followed by an arbitrary number of additions in the target group. The second component is a black-box construction of universally verifiable distributed encryption switching between any public key encryption schemes supporting shared setup and key generation phases, as long as the schemes satisfy some natural additive-homomorphic properties. This allows us to switch back from the target group to the source group, and hence perform an arbitrary number of multiplications. The key generation algorithm of our prototypical cryptosystem, which is based upon concurrent verifiable secret sharing, permits robust re-construction of powers of a shared secret.

1 Introduction

We explore the design of efficient universally verifiable MPC protocols, motivated by applications to the counting of complex ballots in an election. *Universal verifiability* means that the computation should be verifiably correct, even to people who do not participate, and even if all parties involved in the computation are misbehaving. Apart from verifiability, we also require privacy to be guaranteed as long as the number of trustees behaving honestly is above a certain threshold. As trustees must be able to compute the result of the computation, and therefore jointly have access to the inputs, this appears to be the best we can hope for, at least in the absence of extra setup assumptions. (anonymous channel, tamper-proof devices, *etc.*).

Achieving these goals is particularly important in elections: we need the correctness of the tally to be guaranteed, even if all the people in charge of running the election are corrupted – or if all of their computing devices have been hacked – and ballots need to remain secret. Of course, this setting is meaninful in a lot of other contexts: secret bid auctions in which the winning bid is determined by the organisers and, more generally, any cloud application in which a group of users outsource their secret data to one or more cloud service providers, and expect correct computation while maintaining the confidentiality of their data.

Homomorphic encryption lends itself naturally to universally verifiable computation, because the computation itself can be performed by anyone. The private key can be shared among several trustees, who need only prove that they decrypted the final result correctly. For simple elections in which tallying consists only of addition, efficient solutions exist based on additive-homomorphic encryption [19, 2, 7]. We are interested in complex election schemes in which more than a simple sum is needed. Our particular application is *Instant runoff voting (IRV)*, also called alternative voting, which is used in verious places around the globe, either in general public elections (e.g., Australia, Ireland, San Francisco), or in internal consitutencies or political party elections (e.g., Canada, India, U.K.). In IRV, each voter lists some or all the candidates in their order of preference. At each iteration, each ballot is credited towards its highest uneliminated candidate. The candidate with the lowest tally is then eliminated (so each ballot is then credited to its next uneliminated candidate). This terminates when one candidate has a strict majority. This elimination process requires multiplications on top of addition, which cannot be homomorphically achived with traditional efficient schemes like ElGamal or Paillier. For this case, leveled homomorphic encryption [13] would work, but would need to be parameterized in advance for the maximum depth of multiplications that might possibly be needed, and pay an efficiency cost on that basis. In our setting, that depth would be the total number of candidates (minus 2), which might be a lot more than the actual number of eliminations.

1.1 Summary of our contribution

We build a simple universally verifiable MPC protocol from two components.

1. *A somewhat homomorphic encryption scheme with threshold key generation in the malicious static adversary setting.* It is similar to [16] in allowing arbitrary additions in a source space, then one multiplication. Our threshold key generation protocol allows efficient proofs of correct decryption.
2. *A multiparty encryption switching protocol that transforms a ciphertext from the target space, i.e., resulting from a homomorphic multiplication, into a ciphertext in the source space, hence making it possible to perform more multiplications.* This protocol is universally verifiable in the setting of [39].

Our scheme only requires computation in standard prime order groups and relies on standard computational assumptions (e.g., SXDH). The availability of addition and multiplication is sufficient to perform arbitrary computation. It supports threshold key generation in the malicious setting with static corruption.

As a demonstration for our example application, we present a privacy-preserving universally verifiable implementation of the tallying phase of Instant Runoff Voting, based on our universally verifiable computation protocol. Our sample implementation was run on real-world data from public elections in Australia, which

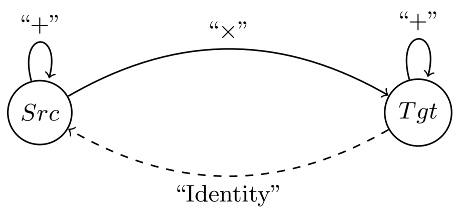

Fig. 1. Operations supported by our encryption scheme. Continuous (resp. dashed) arrows refer to non-interactive (resp. interactive) operations.

shows that our protocol is efficient enough for tallying real-world elections within a reasonable time frame, while leaving ample space for further optimization.

1.2 Comparison with related work on MPC

Our approach bears some resemblance to the encryption-switching approach of Couteau *et al.* [18], but has some significant differences. They switch between additively and multiplicatively homomorphic encryption schemes, while we switch between spaces in which we have additively homomorphic encryption, with the possibility to perform a multiplication as part of a switch. They have two switching protocols, between the additively and multiplicatively homomorphic ciphertext spaces, while we only need a protocol to switch from our target space back to our source space. Their protocols for secure computation are 2-party protocols and highly asymmetric (assigning specific roles to each party), while our protocols are multi-party, perfectly symmetric and universally verifiable.

Catalano and Fiore [16] describe boosting linearly homomorphic encryption to achieve server aided two-party secure function evaluation on parallel inputs in the semi-honest setting. We do not know if this approach can be generalised to the N-party setting. Like [12], their system allows evaluation of 2DNF formulae, that is, an addition, followed by one multiplication, followed by more additions. However, additions in the target space require ciphertext expansion, which is not the case in our scheme.

Three recent works address universally verifiable MPC. Their main bottleneck is key generation. Baum et al. [5] add universally verifiable proofs of correctness to SPDZ [21], which uses a somewhat homomorphic encryption scheme that has n-out-of-n key generation in the covert adversary model. The protocol therefore only offers confidentiality in that model. We have security in the traditional malicious adversary setting. This approach naturally scales to arbitrary multiplications, with cost proportional to the total number actually done. However, the structure of the protocols, based on secret shared data, uses secure bidirectional channels between the input parties (e.g., the voters) and the computing parties (e.g., the election trustees), which is a challenging constraint for large scale applications. Our focus is on single pass protocols [9], in which voters can vote by submitting a single message built from a public election description, and have a computational work independent of the number of trustees.

Schoenmakers and Veeningen [39] rely on Damgaard-Jurik encryption, which supports efficient threshold key generation if an RSA modulus with unknown factorization is available bringing us back to key generation difficulties.

The most closely related work comes from Castagnos et al. [15], who propose new encryption schemes and switching protocols following [18], but working in prime order groups (like we do), hence also supporting threshold operations. They combine additively and multiplicatively homomorphic schemes (while we use a somewhat homomorphic approach). Their encryption scheme however relies on the hardness of DDH in very specific groups: subgroups of the class group of an order of a quadratic field of discriminant $-p^3$, which comes with efficiency penalties. They also need to work in subgroups of unknown order, which increases the cost of the ZK proofs needed for verifiability. Our protocol works in a standard computational setting (traditional asymmetric pairings), with efficiency and compatibility advantages (in particular, standard sigma protocols for prime order groups can be used). The tradeoff between the two would depend on the computation: in our IRV counting setting, we have many additions, followed by a single multiplication, followed by many more additions, repeatedly. For this kind of circuit our approach is more efficient than [15]. However, a computation with unbounded successive multiplications would eventually be faster with their method, despite the use of more expensive components.

In concurrent work, Attrapadung et al. [3] introduce a somewhat homomorphic encryption scheme that is a specific instance of our encryption scheme family. However, they do not offer a threshold (or distributed) variant, or a switching protocol, which are the key ingredients for our universally verifiable MPC protocol, nor do they consider general computation or voting.

1.3 Counting IRV Ballots

Plaintext IRV tallying raises coercion issues. The number of possible votes is more than $c!$ (where c is the number of candidates), which may be much larger than the number of votes actually cast. This introduces the possibility of an attack often called the *Italian attack:* a coercer demands a certain pattern of preferences, presumably with her favourite candidate first, and then checks to see whether that pattern appears in the final tally. To thwart this attack, many works describe universally verifiable IRV tallying without revealing individual ballots [30, 36, 27, 35, 37, 8].

However, these all use mix-nets [33], which count among the most complex cryptographic protocols ever deployed. Besides, even when mixes use strong zero knowledge-proof based verification, if a single mix misbehaves then the entire mix-net halts until a replacement is found, leading to a protocol which is inherently non-robust. Ours is the first universally verifiable scheme for privacy-preserving IRV tallying without mixnets.

For our example application we implemented the single-authority version of our cryptosystem and switching protocol and used it to recount two real IRV elections, using public data from the Australian state of New South Wales. Each election included more than 40,000 ballots. The first, involving 5 candidates and a single elimination round, completed in 2 hours. The second, with 6 candidates and 4 elimination rounds, took 15 hours. This does not include the proofs of correct switching, which would add a constant multiplicative factor The details are in Section J.

1.4 Structure of this paper

The next section contains cryptographic background. In Section 2 we present a new candidate cryptosystem with which to instantiate source and destination encryption schemes for the N-party encryption switching primitive. Next, in Section 3, we tackle the problem of constructing a distributed key generation procedure for this protocol. Then in Section 4 we describe the universally verifiable protocol for switching from target back to source encryption schemes. Our prototype implementation for Instant Runoff Vote counting is in Section 5.

1.5 Background

We define a generic access structure for linear secret sharing schemes.

Definition 1 (Access Structure [32, 41]) *Let $\mathcal{S}$ be a set of parties. A collection $\mathbb{A} \subset 2^{\mathcal{S}}$ is monotone if $\forall\ B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. An access structure, respectively monotone access structure, is a collection (respectively monotone collection) $\mathbb{A}$ of non-empty subsets of $2^{\mathcal{S}}$ i.e., $\mathbb{A} \subseteq 2^{\mathcal{S}} \backslash \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorised sets; the sets not in $\mathbb{A}$ are called unauthorised sets.*

Definition 2 (Linear Secret-Sharing Scheme [6, 41]) *A secret-sharing scheme Π over a set of parties P is called linear over field $\mathbb{Z}_p$ if*

1. *The shares of the parties form a vector of dimension at most l over $\mathbb{Z}_p$.*
2. *There exists a matrix M with ℓ rows and d columns called the share-generating matrix for Π. There also exists a function ρ which maps each row of the matrix to an associated party. That is for $i = 1, \ldots, \ell$, the value $\rho(i)$ is the party associated with row i. When we consider the column vector $v = (s, r_2, \ldots, r_d)^T$, where $s \in \mathbb{Z}_p$ is the secret to be shared, and $r_2, \ldots, r_d \in \mathbb{Z}_p$ are randomly chosen, then Mv is the vector of ℓ shares of the secret s according to Π. The share $(Mv)_i$ belongs to the party $\rho(i)$.*

It is proven in [6] that every every linear secret-sharing scheme (LSSS) satisfies the following property, called *linear-reconstruction* in [41]. Suppose that Π is an LSSS for the access structure $\mathbb{A}$. Let $V \in \mathbb{A}$ be any authorised set, and let $I \subseteq \{1, \ldots, \ell\}$ be defined as $I = \{i : \rho(i) \in V\}$. Then there exist constants $\{\Lambda_{i,V} \in \mathbb{Z}_p : i \in I\}$ such that, if $\{s_i\}$ are valid shares of any secret s according to Π, then $\sum_{i \in I} \Lambda_{i,V} \cdot s_i = s$. Moreover these constants $\{\Lambda_{i,V}\}$ can be found in time polynomial in the dimensions of the share-generating matrix M.

Definition 3 (T-Threshold Access Structure) *Of specific interest for our purposes is the T-party threshold access structure, defined as $\mathbb{A}_{T\text{-Th}} = \{S : S \in 2^{\{P_1,\ldots,P_n\}}, |S| \geq T\}$, where $T < n/2$. Let M be the linear secret-sharing scheme matrix corresponding to $\mathbb{A}_{T\text{-Th}}$. In that case there exists M with row-dimension $l = n$ and column-dimension $d = T$.*

Pairings on Prime-Order Groups To build our one-time homomorphic cryptosystem of Section 3, we require the notion of *projecting bilinear group generators* [25]. Our specific choice of generator will be a variant of the polynomial-induced projecting generator introduced by Herold et al. [31], tailored for the asymmetric pairing setting.

Definition 4 (Bilinear Group Generator [25]) *A bilinear group generator is an algorithm $\mathcal{G}$ that takes as input a security parameter λ and outputs a description of five abelian groups G, G_1, H, H_1, G_t with $G_1 < G$ and $H_1 < H$. Assume that this description permits polynomial-time group operations and random sampling in each group. The algorithm also outputs an efficiently computable map $e : G \times H \to G_t$ that satisfies:*

Bilinearity. *For all $g_1, g_2 \in G$ and $h_1, h_2 \in H$,*
$e(g_1 g_2, h_1 h_2) = e(g_1, h_1)e(g_1, h_2)e(g_2, h_1)e(g_2, h_2)$.
Non-degeneracy. $e(g, h) = 1 \; \forall h \in H \iff g = 1$
and $e(g, h) = 1 \; \forall g \in G \iff h = 1$.

A bilinear group generator $\mathcal{G}$ is prime-order if G, G_1, H, H_1, G_t all have prime order p.

Definition 5 (Projecting Bilinear Group Generator [25]) *Let $\mathcal{G}$ be a bilinear group generator. Say that $\mathcal{G}$ is projecting if it also outputs a group $G'_t < G_t$ and three group homomorphisms π_1, π_2, π_t mapping G, H, G_t to themselves such that*

1. *Subgroups G_1, H_1, G'_t are contained in the kernels of π_1, π_2, π_t respectively.*
2. $e(\pi_1(g), \pi_2(h)) = \pi_t(e(g, h))$ *for all $g \in G, h \in H$.*

We propose a projecting bilinear group operator induced by tensor product, instead of relying on the polynomial product previously proposed [31]. The polynomial solution was designed for the symmetric pairing setting, but raises difficulties in the definition of the projecting operator when moving to the asymmetric setting. Our tensor product based solution offers an efficient alternative that makes it possible to have efficient ciphertext in the base groups, by relying on the sXDH assumption.

Definition 6 (l-Symmetric Cascade Assumption [24]) *Let $\{\mathbb{G}_\lambda\}_\lambda$ be an ensemble of cyclic groups with prime-orders $\{\mathbb{Z}_{p(\lambda)}\}_\lambda$ where $\exists c > 0 \; \forall \lambda \; |p(\lambda)| < \lambda^c$.*

For fixed λ, let $\mathbb{Z}_p = \mathbb{Z}_{p(\lambda)}$ and define the distribution of matrices over $\mathbb{Z}_p^{(l+1)\times l}$:

$$\mathcal{SC}_l =: \begin{pmatrix} -s & 0 & \dots & 0 & 0 \\ 1 & -s & \dots & 0 & 0 \\ 0 & 1 & & 0 & 0 \\ & \ddots & & \ddots & \\ 0 & 0 & \dots & 1 & -s \\ 0 & 0 & \dots & 0 & 1 \end{pmatrix} : s \in_R \mathbb{Z}_p.$$

Then $\forall$ PPT adversaries $\mathcal{A}$, the difference below is a negligible function of λ.

$$|\Pr[1 \leftarrow \mathcal{A}(\mathbb{G}, g, g^A, g^{A\boldsymbol{w}}) : g \in_R \mathbb{G}, A \in \mathcal{SC}_l, \boldsymbol{w} \in_R \mathbb{Z}_p^l] -$$
$$\Pr[1 \leftarrow \mathcal{A}(\mathbb{G}, g, g^A, g^{\boldsymbol{u}}) : g \in_R \mathbb{G}, A \in \mathcal{SC}_l, \boldsymbol{u} \in_R \mathbb{Z}_p^{l+1}]|.$$

Definition 7 (External l-Symmetric Cascade Assumption) *Let $\mathcal{D}_1, \mathcal{D}_2$ and $\mathcal{D}_t$ be three ensembles of cyclic groups, such that for every $\lambda \in \mathbb{N}$, if $\mathbb{G}_1 = \mathbb{G}_{1\lambda} \in \mathcal{D}_1$, $\mathbb{G}_2 = \mathbb{G}_{2\lambda} \in \mathcal{D}_2$ and $\mathbb{G}_t = \mathbb{G}_{t\lambda} \in \mathcal{D}_t$, there exists an efficiently computable pairing $e(\cdot,\cdot)$, such that $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$. The External l-Symmetric Cascade assumption is that the l-Symmetric Cascade assumption holds in each of the ensembles $\mathcal{D}_1$ and $\mathcal{D}_2$.*

Proposition 1 *The Symmetric External Diffie-Hellman Assumption [40] holds with respect to group ensembles $\mathcal{D}_1, \mathcal{D}_2$, iff the External 1-Symmetric Cascade Assumption holds with respect to $\mathcal{D}_1, \mathcal{D}_2$.*

CF Encryption Recently Catalano and Fiore [16] showed how to generalise earlier work on 2DNF formulae [12] to transform virtually any linearly homomorphic cryptosystem into one permitting the computation of any degree-2 formula. Multiplication transforms two input ciphertexts from a "level-1" space into an encryption of the product in the "level-2" space. In this level-2 space, further homomorphic additions remain possible, at the cost of ciphertext expansion at each step. Still, it is not possible to perform any further multiplications.

For concreteness we will assume additive ElGamal encryption for the base public key encryption scheme. Let $(\overline{\mathsf{Keygen}}, \overline{\mathsf{Enc}}, \overline{\mathsf{Dec}})$ be additive ElGamal on message space $(\mathcal{M}, +)$. The Catalano-Fiore cryptosystem is as follows.

$\mathsf{Keygen}(1^\lambda)$ Let $(\overline{\mathsf{pk}}, \overline{\mathsf{sk}}) \leftarrow \overline{\mathsf{Keygen}}(1^\lambda)$.
  Set $(\mathsf{pk}, \mathsf{sk}) \leftarrow (\overline{\mathsf{pk}}, \overline{\mathsf{sk}})$.

$\mathsf{Enc}(\mathsf{pk}, M)$ Choose $b \in_R \mathcal{M}$.
  Output $C = (M - b, \overline{\mathsf{Enc}}(\mathsf{pk}, b))$.

$\mathsf{Multiply}(\mathsf{pk}, C, C')$ Let $C = (C_0, C_1)$ and $C' = (C'_0, C'_1)$ be inputs. Let $\alpha = \overline{\mathsf{Enc}}(\overline{\mathsf{pk}}, C_0 C'_0) \cdot (C_1)^{C'_0} \cdot (C'_1)^{C_0}$.
  Output (α, C_1, C'_1).

$\mathsf{Dec}(\mathsf{sk}, C)$ Accept $C = (\alpha, C_1, C'_1)$ as input.
  Let $M' \leftarrow \overline{\mathsf{Dec}}(\overline{\mathsf{sk}}, \alpha)$, $b \leftarrow \overline{\mathsf{Dec}}(\overline{\mathsf{sk}}, C_1)$ and
  $b' \leftarrow \overline{\mathsf{Dec}}(\overline{\mathsf{sk}}, C'_1)$ as input. Output $M = M' + bb'$.

Noninteractive Zero Knowledge Proofs We use non-interactive zero knowledge proofs of the following NP relations. Efficient constructions of these can be found in Appendix D. Let $\mathit{\Pi}_{\mathsf{range}} = (G_{\mathsf{range}}, P_{\mathsf{range}}, V_{\mathsf{range}})$ be a non-interactive zero knowledge proof for the relation $\mathcal{R}_{\mathsf{range}} = \{(c, y) | \exists\ a, r : c_i = \mathsf{Enc}(y, a; r) \wedge a \in [0, 2^{\lambda} - 1]\}$. Let $\mathcal{R}_{\mathsf{bit}} \subseteq \mathcal{R}_{\mathsf{range}}$ be the special case $\lambda = 1$ and $\mathit{\Pi}_{\mathsf{bit}}$ be the corresponding proof system. Let $\mathit{\Pi}_{\mathsf{eq}} = (G_{\mathsf{eq}}, P_{\mathsf{eq}}, V_{\mathsf{eq}})$ be a non-interactive zero knowledge proof system for the relation $\mathcal{R}_{\mathsf{eq}} = \{(c, c', \mathsf{pk}_1, \mathsf{pk}_2) | \exists m, r, r' : c = \mathsf{Enc}_1(\mathsf{pk}_1, m; r) \wedge c' = \mathsf{Enc}_2(\mathsf{pk}_2, m; r')\}$. For $1 \leq j \leq N$ let σ_j be the common reference string belonging to P_j.

2 One-time Multiplicatively Homomorphic Cryptosystem

The basis of our universally verifiable MPC protocol is a homomorphic cryptosystem that supports arbitrarily many additions, followed by one multiplication, followed by arbitrarily many additions.

Many such encryption schemes have been already proposed, starting with the BGN pairing-based scheme [12]. However, threshold key generation for BGN and similar schemes is challenging, as it would require the generation of RSA-type moduli with unknown factorization, and computing in the resulting pairing groups of composite order is also quite demanding. Unverifiable trust assumptions would undermine the main purpose of this work.

This motivates our construction of a pairing based homomorphic cryptosystem on *prime-order* groups, for which a secure and robust key generation procedure can be derived. This has been explored by Freeman [25], who shows how to build such schemes from projecting pairings and, more recently by Herold et al. [31] who show how to build them from hidden matrix-rank based indistinguishability assumptions [24] on the source group of symmetric pairings.

As these symmetric pairings have also become extremely expensive from a computational point of view due to the recent attacks on the discrete logarithm in low characteristic, we aim for a more efficient scheme based an asymmetric pairings, by extending their work to that setting. This requires performing operations in parallel in the two source groups of the pairing, and designing a tensor product-based projecting pairing as a replacement for their polynomial product. The underlying indistinguishability problem induced by this pairing on both source groups is a generalisation of the well-known XDH problem [40].

This section contains only the simplest instance of our encryption scheme, based on the External 1-Symmetric Cascade Assumption. A general version based on the External l-Symmetric Cascade Assumption is presented in Appendix C. We first construct a projecting bilinear group as a special case of Definition 15 (in Appendix C) with $l = 1$.

Definition 8 (Projecting pairing construction) *Take as input a prime-order bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \hat{e})$, elements $g \in \mathbb{G}_1$ and $h \in \mathbb{G}_2$, and secret keys s and s' in $\mathbb{Z}_p$.*

Define $G = \mathbb{G}_1^2$, $H = \mathbb{G}_2^2$, $G_t = \mathbb{G}_t^4$, and define the bilinear map $e : G \times H \to G_t$ as $e((g_0, g_1), (h_0, h_1)) = (\hat{e}(g_0, h_0), \hat{e}(g_0, h_1), \hat{e}(g_1, h_0), \hat{e}(g_1, h_1))$.

Define G_1 (resp. H_1) as the subgroup of G (resp. H) generated by $g^{(-s,1)} = (g^{-s}, g)$ (resp. $h^{(-s',1)}$).

Define the following projecting maps:

– $\pi_1 : G \to \mathbb{G}_1$ *as* $\pi_1(g_1, g_2) = g_1 g_2^s$,

– $\pi_2 : H \to \mathbb{G}_2$ *as* $\pi_2(h_1, h_2) = h_1 h_2^{s'}$,

– $\pi_t : G_t \to \mathbb{G}_t$ *as* $\pi_t(g_1, g_2, g_3, g_4) = g_1 g_2^{s'} g_3^s g_4^{ss'}$.

Output secret key $\mathsf{sk} = (\pi_1, \pi_2, \pi_t)$ *and public key* $\mathsf{pk} = (G, G_1, H, H_1, G_t, e, g, h)$.

It is easy to see that G_1 and H_1 are the kernel of π_1 and π_2 and that these operators essentially offer a decryption operation for ElGamal-like encryption schemes that use s and s' as secret keys.

Notation: $\boldsymbol{v_1} \cdot \boldsymbol{v_2}$ denotes elementwise multiplication; $\boldsymbol{v_2}^n$ is elementwise exponentiation.

Our encryption scheme is then defined as follows.

$\mathsf{Setup}(1^\lambda)$: Let $\mathcal{P}$ be a prime-order bilinear group generator. Let $\mathcal{M} = \mathbb{Z}_p$. Output $\mathsf{pp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \hat{e}) \leftarrow \mathcal{P}(1^\lambda)$.

$\mathsf{KeyGen}(\mathsf{pp})$: Select s and s' in $\mathbb{Z}_p$, set $\boldsymbol{x} = (-s, 1)$ and $\boldsymbol{x'} = (-s', 1)$. Choose $g \in_R \mathbb{G}_1, h \in_R \mathbb{G}_2$, and define $\mathbf{g} = g^{\boldsymbol{x}} = (g^{-s}, g)$ and $\mathbf{h} = h^{\boldsymbol{x}'} = (h^{-s'}, h)$. Run the Projecting Pairing construction on input pp, g, h, s, s'. Output the resulting secret key $\mathsf{sk} = (\pi_1, \pi_2, \pi_t)$ and the public key $\mathsf{pk} = (G, G_1, H, H_1, G_t, e, g, h)$. Note that G_1 and H_1 are described by their generators $\mathbf{g}$ and $\mathbf{h}$ respectively.

$\mathsf{Enc}_{\mathsf{src}}(\mathsf{pk}, M)$: Choose a, b at random in $\mathbb{Z}_p$. Let $\mathbf{g}_1 = (\mathbf{g})^a = (g^{-as}, g^a)$ and $\mathbf{h}_1 = (\mathbf{h})^b = (h^{-bs'}, h^b)$. Let $C_0 = \mathbf{g}^M \cdot \mathbf{g}_1, C_1 = \mathbf{h}^M \cdot \mathbf{h}_1$. Output the ciphertext (C_0, C_1) in $G \times H$.

$\mathsf{Enc}_{\mathsf{tgt}}(\mathsf{pk}, M)$: Choose a, b at random in $\mathbb{Z}_p$. Let $\mathbf{g}_1 = (\mathbf{g})^a = (g^{-as}, g^a)$ and $\mathbf{h}_1 = (\mathbf{h})^b = (h^{-bs'}, h^b)$. Output the ciphertext $C = e(\mathbf{g}, \mathbf{h})^M \cdot e(\mathbf{g}, \mathbf{h}_1) \cdot e(\mathbf{g}_1, \mathbf{h})$ in G_t.

$\mathsf{Multiply}_{\mathsf{src}}(\mathsf{pk}, C, C')$: Take as input two ciphertexts $C = (C_0, C_1)$ and $C' = (C'_0, C'_1)$. Choose $\mathbf{g}_1 \in_R G_1$ and $\mathbf{h}_1 \in_R H_1$, as in the above routine. Output $C = e(C_0, C'_1) \cdot e(\mathbf{g}, \mathbf{h}_1) \cdot e(\mathbf{g}_1, \mathbf{h})$, an element of G_t.

$\mathsf{Add}_{\mathsf{src}}(\mathsf{pk}, C, C')$: Take as input two ciphertexts $C = (C_0, C_1)$ and $C' = (C'_0, C'_1)$. Choose $\mathbf{g}_1 \in_R G_1$ and $\mathbf{h}_1 \in_R H_1$. Let $C''_0 = C_0 \cdot C'_0 \cdot \mathbf{g}_1$. Let $C''_1 = C_1 \cdot C'_1 \cdot \mathbf{h}_1$. Output $C'' = (C''_0, C''_1)$.

$\mathsf{Add}_{\mathsf{tgt}}(\mathsf{pk}, C, C')$: Take as input two ciphertexts C and C' in G_t. Choose $\mathbf{g_1} \in_R G_1$ and $\mathbf{h}_1 \in_R H_1$.
Let $C'' = C \cdot C' \cdot e(\mathbf{g}, \mathbf{h}_1) \cdot e(\mathbf{g}_1, \mathbf{h})$. Output C''.

$\mathsf{Dec}_{\mathsf{src}}(\mathsf{sk}, C)$: Take as input a ciphertext $C = (C_0, C_1)$ in $G \times H$. Compute $M \leftarrow \log_{\pi_1(\mathbf{g})}(\pi_1(C_0))$ and $M' \leftarrow \log_{\pi_2(\mathbf{h})}(\pi_2(C_1))$. Output M if $M = M'$ or $\perp$ otherwise.

$\mathsf{Dec}_{\mathsf{tgt}}(\mathsf{sk}, C)$: Take as input a ciphertext C in G_t. Output $M \leftarrow \log_{\pi_t(e(\mathbf{g}, \mathbf{h}))}(\pi_t(C))$.

Lemma 1. *Suppose that the External 1-Symmetric Cascade assumption, i.e., Symmetric External Diffie Hellman assumption, holds with respect to the groups $\mathbb{G}_1$ and $\mathbb{G}_2$. Then the above cryptosystem is semantically secure.*

Proof. See Appendix C.

3 Distributed Key Generation Protocol for One-time Multiplicative Homomorphic Cryptosystem

In this section we describe key generation for the one-time multiplicatively homomorphic cryptosystem of Section 2. Traditional protocols for threshold key generation [34, 26] would be a natural choice, except that they fail for the $\mathsf{Dec}_{\mathsf{tgt}}$ algorithm, because the evaluation of π_t requires the sharing of a quadratic secret ss', while the traditional protocols are defined for linear terms only.

To overcome this difficulty, our protocol requires each party in the qualified set to split their individual secrets into chunks over a small interval. We construct a blinded version, i.e, $ss' + b$, in which the blinding factor b is distributed across parties, in such a way that it can be cancelled out from shares submitted by a qualified set. To perform the private construction of the blinded square, we use the Catalano-Fiore transformation [16], which enables depth-one multiplications on any linearly homomorphic cryptosystem. A problem arises with the natural choice of additive El Gamal as the base scheme with which to bootstrap the computation of the square. This cryptosystem mandates that only secrets from a small space can be safely decrypted, while the space over which s and s' are derived is much larger. We solve this problem by splitting the individual secrets of qualified players into chunks. Thus the private product of individual secrets becomes equivalent to a private product of polynomials, crucially ones for which the coefficient space is small and therefore amenable to the discrete log problem.

Another problem is how to construct the blinding factor so that no information is leaked on ss' in the construction of $ss' + b$. We show that this is possible via direct verifiable secret sharing of the chunks corresponding to b in polynomial form. As long as the chunk-size used to derive b is sufficiently larger than the chunk-size used to derive ss', we may treat them as distinct secrets to be jointly constructed by the qualified set. For this, and for constructing the Catalano-Fiore encryption key, we may simply employ the key-generation protocol of Pedersen [34] or the later protocol by Gennaro *et al.* [26].

Thus, after CF decryption, a blinding of the square of the secret is revealed in the clear, while the blinding factor is a distributed secret. The blinding factor can be cancelled out "on demand" by a threshold set of qualified players, leading to a fully contained key generation protocol for our multiplicative cryptosystem. Like the key generation protocols of [34, 17, 26], our protocol uses concurrent verifiable secret sharing to build a secret key but assumes as input shares of a transport key under which the main key generation protocol runs. For the latter purpose one may use any of those schemes.

Let $[\cdot]_y$ denote a CF encryption under key y. Let $g_1, g_2, g_{\mathsf{vss}}, g_{\mathsf{pke}} \in \mathbb{G}_1$ and $h_1, h_2, h_{\mathsf{pke}} \in \mathbb{G}_2$ be public. Let $c_A = 2^{\lambda_A}$ and $c_B = 2^{\lambda_B}$ be the chunk sizes of individual secrets and individual blinding factors. One may set $c_A = p^{\frac{1}{4l}} \cdot 2^{-\frac{\lambda}{2}}$ and $c_B = p^{\frac{1}{2l}}$ where l is chosen so that discrete logarithms are feasible in the range $[0, N \cdot p^{\frac{1}{2l}}]$. Appropriate sizes are given in Lemma 3, Appendix F.

Recall the security properites of a distributed key generation protocol [26].

Correctness : All subsets of T shares provided by honest players define the same unique secret key sk; all honest parties have the same value of the public key pk, which is correct wrt sk; sk is uniformly distributed among a range $\{0,1\}^\lambda$, where λ is the security parameter.

Resilience : There is a procedure to reconstruct the secret key sk out of T or more shares, which is resilient in the presence of malicious parties.

Security : No information can be learned on sk except for what is implied by the public key pk.

The full protocol is given in Figure 2. The NIZKs are described in Appendix D.

Protocol 1: Key Generation for one-time homomorphic cryptosystem

Common Input : CF public key y. Generators $g_{\mathsf{vss}}, g_{\mathsf{pke}} \in \mathbb{G}_1$ and $h_{\mathsf{pke}} \in \mathbb{G}_2$. Chunk sizes $c_A = 2^{\lambda_A}$ and $c_B = 2^{\lambda_B}$, of individual secrets and individual blinding factors respectively.

Private Input : P_i holds shares s_i, s'_i and t_i of secret keys s, s' and t respectively.

Public Output : Public key pk for the source and target encryption schemes.

Private Output : To each P_i, shares x_i and x'_i of the source and target encryption schemes, and shares b_i of $ss' + b$. Blinding factor γ.

1. *Each party P_i breaks its secret shares into chunks, commits publicly to the chunks, and shares individual chunks with other parties as follows.* Write s_i as $\sum_{k=0}^{\ell-1} \alpha_{ik} c_B^k$, s'_i as $\sum_{k=0}^{\ell-1} \alpha'_{ik} c_B^k$, and t_i as $\sum_{k=0}^{2\ell-2} \beta_{ik} c_B^k$, where $\alpha_{ik}, \alpha'_{ij} \in_R [0, 2^{\lambda_A} - 1]$ and $\beta_{ik} \in_R [0, 2^{\lambda_B} - 1]$. P_i creates vectors $\boldsymbol{v}_i = (s_i, r_{i2}, \ldots, r_{iT})^t, \boldsymbol{v}'_i = (s'_i, r'_{i2}, \ldots, r'_{iT})^t, \boldsymbol{w}_i = (t_i, r''_{i2}, \ldots, r''_{iT})^t$. Recall secret-sharing matrix M from Definition 3. P_i computes the share vectors $\boldsymbol{s}_i = M\boldsymbol{v}_i, \boldsymbol{s}'_i = M\boldsymbol{v}'_i$ and $\boldsymbol{t}_i = M\boldsymbol{w}_i$. Let $V_i = g_{\mathsf{vss}}^{\boldsymbol{v}_i}, V'_i = g_{\mathsf{vss}}^{\boldsymbol{v}'_i}, W_i = g_{\mathsf{vss}}^{\boldsymbol{w}_i}$. P_i broadcasts the values $\{V_i, V'_i, W_i\}$. P_i sends $s_{ij} = \boldsymbol{s}_i[j], s'_{ij} = \boldsymbol{s}'_i[j], t_{ij} = \boldsymbol{t}_i[j]$ to each P_j via a private channel, for $1 \le j \le N$. Note that
$$g_{\mathsf{vss}}^{s_{ij}} = V_i^{M(j)}, g_{\mathsf{vss}}^{s'_{ij}} = {V'_i}^{M(j)}, g_{\mathsf{vss}}^{t_{ij}} = W_i^{M(j)} \tag{1}$$
2. P_i verifies that the shares received from P_j, i.e., s_{ji}, s'_{ji} and t_{ji} are correct, by verifying Equation 1. If any of these equations do not hold for the received values s_{ji}, s'_{ji} and t_{ji}, P_i broadcasts the message $(P_i, \mathsf{complain}, P_j)$.
3. For each broadcast message $(P_{i_\alpha}, \mathsf{complain}, P_j)$, player P_j is disqualified if $(s_{ji_\alpha}, s'_{ji_\alpha}, t_{ji_\alpha})$ are sent that do not satisfy Equation 1. Let Q be the set of continuing (i.e. non-disqualified) players.

Fig. 2. Key gen protocol for one-time homomorphic cryptosystem.

Protocol 1– Part 2

4) *Each party commits to its blinding factors and share vectors from Step 1, then proves that the chunks it shared in Step 1 are within the required range, and that the chunks sum correctly to the committed values, as follows.*

   Let $A_i = g_{\mathsf{pke}}^{\boldsymbol{v}_i}, B_i = g_{\mathsf{pke}}^{\boldsymbol{w}_i}, A'_i = h_{\mathsf{pke}}^{\boldsymbol{v}'_i}, B'_i = h_{\mathsf{pke}}^{\boldsymbol{w}_i}$, $C_i = ([\alpha_{i0}]_y, \ldots, [\alpha_{i(\ell-1)}]_y)$, $C'_i = ([\alpha'_{i0}]_y, \ldots, [\alpha'_{i(\ell-1)}]_y)$, $D_i = ([\beta_{i0}]_y, \ldots, [\beta_{i(2(\ell-1))}]_y)$ where $\boldsymbol{v}_i, \boldsymbol{v}'_i, \boldsymbol{w}_i$ are sampled as in Step 1. Let $\varepsilon_i \leftarrow (P_{\mathsf{range}}((C_{ik})_k, c_A)$, $P_{\mathsf{range}}((C'_{ik})_k, c_A)$, $P_{\mathsf{range}}((D_{ik})_k, c_B)$, $P_{\mathsf{eq}}(A_i[1], \prod_{k=0}^{\ell-1}[\alpha_{ik}]_y^{c_B^k})$, $P_{\mathsf{eq}}(A'_i[1], \prod_{k=0}^{\ell-1}[\alpha'_{ik}]_y^{c_B^k})$, $P_{\mathsf{eq}}(B_i[1], \prod_{k=0}^{2(\ell-1)}[\beta_{ik}]_y^{c_B^k}))$. P_i broadcasts the values $\{A_i, A'_i, B_i, B'_i, C_i, C'_i, D_i, \varepsilon_i\}$. Note that

$$g_{\mathsf{pke}}^{s_{ij}} = A_i^{M(j)}, g_{\mathsf{pke}}^{t_{ij}} = B_i^{M(j)},$$
$$h_{\mathsf{pke}}^{s'_{ij}} = A_i'^{M(j)}, h_{\mathsf{pke}}^{t_{ij}} = B_i'^{M(j)} \qquad (2)$$

5) P_i verifies that for the values sent by every other P_j in Q, Equation 2 holds. If any of these equations do not hold for the values s_{ji}, s'_{ji} and t_{ji}, P_i broadcasts the message $(P_i, \mathsf{complain}, P_j)$.
6) For each broadcast message $(P_{i_\alpha}, \mathsf{complain}, P_j)$ or proofs not satisfying $V_{\mathsf{range}}(\sigma_j, (C_j, C'_j, D_j), \varepsilon_j) = 1 \wedge V_{\mathsf{eq}}(\sigma_j, (A_j, A'_j, B_j), (C_j, C'_j, D_j), \varepsilon_j) = 1$ the other players in Q reconstruct the values $s_j, t_j, \boldsymbol{v}_j, \boldsymbol{w}_j, A_j, A'_j, B_j, B'_j, C_j, C'_j, D_j$.
7) For $0 \leq k \leq 2(\ell-1)$, P_i computes $\mathsf{ct}_k = \sum_{i,j\in Q}\sum_{f+g=k} C_{if}C'_{jg} + \sum_{i\in Q} D_{ik}$ and $\gamma_k \leftarrow \mathsf{Dec}(\mathsf{k}_i, \mathsf{ct}_k)$. Outputs $\gamma = \sum_{k=0}^{2(\ell-1)} \gamma_k c_B^k$.
8) P_i computes their share of the secret as the sum of all shares received in Step 2 among continuing players, i.e., $x_i = \sum_{j\in Q} s_{ji}$ $x'_i = \sum_{j\in Q} s'_{ji}$ and $b_i = \sum_{j\in Q} t_{ji}$. P_i computes $\mathsf{vk}_i = (g_{\mathsf{vss}}^{x_i}, g_{\mathsf{vss}}^{x'_i}, g_{\mathsf{vss}}^{b_i})$ and $y_{\mathsf{pke}} = \prod_{i\in Q} A_i[1], z_{\mathsf{pke}} = \prod_{i\in Q} A'_i[1]$. P_i sets $\mathbf{g_1} = (y_{\mathsf{pke}}, g_{\mathsf{pke}})$ and $\mathbf{h_1} = (z_{\mathsf{pke}}, h_{\mathsf{pke}})$. The public key is $\mathsf{pk} = ((g_1, g_2), \mathbf{g_1}, (h_1, h_2), \mathbf{h_1}, \{[\gamma_k]_y\}_k, \{V_i, V'_i, W_i\}_{i\in Q})$. The secret is (x, x', b, γ). Note that x, x' and b are distributed secrets while γ is held in entirety by each player in Q.

Fig. 3. Key gen protocol for one-time homomorphic cryptosystem, Part 2.

3.1 Protocol description and security properties

Theorem 2. *Protocol 1 is a distributed key generation protocol for the cryptosystem of Section 3 and that is correct, resilient and secure against an active adversary corrupting fewer than T statically chosen players.*

Proof. See Appendix K.

Proposition 3 *The values $x = \sum_{i \in Q} s_i$, $x' = \sum_{i \in Q} s'_i$ and $b = \sum_{i \in Q} t_i$ are distributed secrets according to the threshold access structure.*

Proof. See Appendix K.

Proposition 4 *The values γ, x, x' and b computed in Step 6 satisfy the relation $\gamma = xx' + b$.*

Proof. See Appendix K.

4 Distributed Encryption Switching

In this section we present universally verifiable switching between target and source encryption schemes using only the additive homomorphism on the ciphertext spaces. The protocol is in Figure 4. The idea is for each party to contribute an equivalent encryption of a blinding factor under both cryptosystems together with a zero knowledge proof of plaintext equality. In the source space the blinding factors are homomorphically added to the input ciphertext and the result decrypted under a threshold decryption scheme. From this plaintext, the blinding factors under the target encryption scheme are homomorphically subtracted, producing an encryption of the input message under the target cryptosystem.

To blind the ciphertexts without increasing the size of the messages (remember that it requires a DL extraction), we apply the blinding using an xor-sum. Specifically, we assume an ideal functionality for bit-wise sum, $\mathcal{F}_{\mathsf{SUM}}$ with the following behaviour:

- On input $(\mathsf{setup}, 1^\lambda)$ initialises $\mathcal{D} \leftarrow \emptyset, t \leftarrow 0$.
- On input (send, C), if $t < N$, sets $\mathcal{D} \leftarrow \mathcal{D} \cup \{C\}, t \leftarrow t + 1$, if $t = N$, output C_s which is an encryption of the bit-wise sum of all decrypted ciphertexts contained in $\mathcal{D}$.

The details of the protocol realising this functionality are in Appendix E.

If the ciphertexts are known to be small, the xor-sum can be avoided and we can just homomorphically add a blinding factor, like we did for key generation. This blinding factor can be large enough to offer statistical blinding (e.g., 40 bits more than an upper-bound on the plaintext size) and small enough to support efficient decryption, possibly using a baby-step giant-step algorithm. This comes with the benefit of being a completely non interactive process, and works fine for our voting application.

Our definition of universally verifiable secure computation is derived from [39] and given in Appendix H. It formalises the idea that either a threshold of honest participants produces a true answer, or the output fails verification.

Theorem 5. *Protocol π_{SWITCH} securely computes universally verifiable encryption switching in the $\mathcal{F}_{\mathsf{SUM}}$-hybrid model against statically chosen adversaries if π_{COM} is a secure non-malleable commitment scheme and $\mathcal{P}_{\mathsf{eq}}$ is a secure NIZK proof system.*

Proof. See Appendix K.

Given that the switch is the only operation of our protocols that requires the use of secret information (i.e., decryption keys), and that this operation is verifiable, we obtain a universally verifiable MPC protocol: addition and multiplication are publicly performed using our encryption scheme, and the verifiable switch offers the possibility to repeat these operations as often as needed. In Appendix H.2 we use this approach to evaluate any function class representable by an arithmetic circuit of polynomial size over $\mathcal{M}$.

Protocol π_{SWITCH} for Player P_i.
Common Input : $c = \mathsf{Enc}_1(\mathsf{pk}, m) : m \in \mathcal{M}$ and π_{COM} be a non-malleable commitment scheme with key ck. Threshold t.
Private Input : P_i holds a share of the secret key, sk_i

1. Choose $u_i \in_R \mathbb{Z}_p$ and publish $\delta_i = \mathsf{com}_{ck}(u_i)$ using randomiser r_i.
2. Publish $C'_i = \mathsf{Enc}_1(\mathsf{pk}, u_i)$ and $\overline{C}_i = \mathsf{Enc}_2(\mathsf{pk}, u_i)$ and $\varepsilon_i \leftarrow (\mathcal{P}_{\mathsf{eq}}(\delta_i, C'_i), \mathcal{P}_{\mathsf{eq}}(C'_i, \overline{C}_i))$.
3. If at least t of the ε_i pass verification, let $C' = \prod_{j=1}^{\lambda} c'_{ij} \otimes 2^{j-1}$ and $\overline{C} = \prod_{j=1}^{\lambda} \overline{c}_{ij} \otimes 2^{j-1}$ where $(c'_{ij})_{j=1}^{\lambda} \leftarrow \pi_{\mathsf{SUM}}(C'_1, \ldots, C'_N)$ and $(\overline{c}_{ij})_{j=1}^{\lambda} \leftarrow \pi_{\mathsf{SUM}}(\overline{C}_1, \ldots, \overline{C}_N)$. Otherwise output $\perp$.
4. Let $d \leftarrow c \cdot C', d_i \leftarrow d^{\mathsf{sk}_i}, \xi_i \leftarrow \Sigma_{\mathsf{CD}}(d, d_i, \mathsf{pk}, \mathsf{vk}_i)$.
5. If at least t pass verification for both ε_i and ξ_i, let $m' \leftarrow \prod_{i=1}^{T} d_i$ and output $\overline{c} = \mathsf{Enc}_2^*(m') \cdot \overline{C}^{-1}$. Otherwise output $\perp$.

Fig. 4. Protocol π_{SWITCH}.

5 Tallying Instant Runoff Voting (IRV)

In this section we describe how to use the primitives described earlier to construct a universally verifiable protocol for tallying encrypted ballots according to the IRV algorithm. Ballots are input to the tallying protocol in encrypted form. We reveal only the tallies of each candidate after each round of the IRV algorithm. The main challenge is to ensure that the privacy of ballots is maintained between tallying rounds. We use distributed encryption switching on the cryptosystems $\mathit{\Pi}_{\mathsf{src}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_{\mathsf{src}}, \mathsf{Dec}_{\mathsf{src}})$ and $\mathit{\Pi}_{\mathsf{tgt}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_{\mathsf{tgt}}, \mathsf{Dec}_{\mathsf{tgt}})$ of Section C. Suppose that $\mathit{\Pi}_{\mathsf{tgt}} \rightarrow \mathit{\Pi}_{\mathsf{src}}$ is a distributed encryption switching protocol, where $\mathsf{Enc}_{\mathsf{src}}$ is used to encrypt votes. Recall that in an IRV election, after each phase of tallying, if a candidate is not elected, then the candidate with fewest votes is eliminated. Each ballot should count towards its most-preferred uneliminated candidate. We can use the one-time multiplicative homomorphism to compute the necessary product computations on ballots for the first two rounds of tallying. This takes ballots from the ciphertext space of Π_{src} to the ciphertext space of Π_{tgt}, for which addition, but not multiplication, is possible. To compute the product computations corresponding to further rounds of tallying, the election trustees will come together and perform a distributed switch on the ballots, will take them back to the ciphertext space of Π_{src}, and for which multiplications are again possible. In this way, for every round of tallying after the first, distributed encryption switching can be used to ensure that the trustees can compute the tally for each uneliminated candidate.

5.1 Protocol Details

Ballot representation. Assume c candidates and M voters. An IRV ballot allows expression of up to k preferences, where $k \leq c$ is a constant specific to the election. For the purpose of homomorphic tallying, we will use a special "preference-order" ballot. Let $\mu_n : \{1, \ldots, k\} \to \{1, \ldots, c\}$ be an (injective) function representing the preferences of voter n. The ballot used for tallying, B_n, will be an encryption of the indicator vectors $\mathbf{e}_{\mu_n(1)}, \ldots, \mathbf{e}_{\mu_n(k)}$. The indicator vector $\mathbf{e}_{\mu_n(j)}$ is encrypted as a tuple of c ciphertexts, $\mathbf{v}_j$. Thus B_n is simply a list of k encrypted c-tuples Figure 5 (left) shows an example.
Updating of ballots. This ballot representation permits a convenient method for eliminating candidates, by simply striking out the corresponding column in B_n's matrix of preferences. Since each elimination is a function of publicly verifiable totals, there is no ambiguity as to the representation of any ballot at any stage of tallying. An important feature of this is that the sequence of accesses made by Protocol 2 is derivable from the sequence of intermediate tallies it produces until termination. Input obliviousness follows. Figure 5 (right) shows a preference-order ballot after a candidate has been eliminated.

preference\ candidate	1 2 3 4 5 6		1 2 3 4 5 6
1	0 0 1 0 0 0		0 0 × 0 0 0
2	0 0 0 0 1 0		0 0 × 0 1 0
3	1 0 0 0 0 0		1 0 × 0 0 0

Fig. 5. Preference-order ballot for $c = 6$ and $k = 3$, in its initial form (left) and after elimination of candidate 3 (right), when it should count in candidate 5's tally.

Tallying votes. Let $B_n = (\mathbf{v}_1, \ldots, \mathbf{v}_k)$ be a ballot, S_{C} be the set of uneliminated candidates, and $\Sigma_{S_{\mathrm{C}}}(\mathbf{v}_i)$ be the homomorphic sum of the entries of the i^{th} preference vector over uneliminated candidates. Clearly $\Sigma_{S_{\mathrm{C}}}(\mathbf{v}_i)$ is an encryption of 1 iff the i^{th} preference is for an uneliminated candidate, and an encryption of 0 otherwise. Let $C \boxtimes_{\mathsf{src}} C' = \mathsf{Enc}_{\mathsf{src}}(\mathsf{pk}, MM') : M = \mathsf{Dec}_{\mathsf{src}}(\mathsf{sk}, C)$ and $M' = \mathsf{Dec}_{\mathsf{src}}(\mathsf{sk}, C')$. After $l \leq k$ rounds of tallying, the product

$$\boldsymbol{\pi}_j^{(l)} :=_{1 \leq j' \leq j}^{\boxtimes_{\mathsf{src}}} (\mathsf{Enc}_1^*(1) - \Sigma_{S_\mathrm{C}}(\mathbf{v}_{j'})) : j \leq l$$

is an encryption of 0 iff at least one of the first j preferences is for an uneliminated candidate, and an encryption of 1 otherwise. After $l-1$ rounds of tallying, there is at least one $j \leq l$ such that the j^{th} preference is for a continuing candidate.[4] Therefore after l rounds of tallying, the homomorphic dot product $\sum_{j=1}^{l} \mathbf{v}_j \boxtimes_{\mathsf{src}} \boldsymbol{\pi}_j$ is an encryption of the indicator vector describing which candidate this vote should count for in round l. The protocol is shown in Figure 14, Appendix J.

Implementation We implemented the single-authority version of our system and tested it using elections data for the districts of Albury and Auburn for the 2015 New South Wales state election.[5] The implementation encrypted each of the entries in the ballot matrix prior to commencing the count, to simulate the receipt of encrypted ballots. Ballots were represented as per Figure 5. The experiments were performed on an Intel i7-6770HQ with 4 cores (8 threads) and 32GB RAM. The results are shown in Table 1.

We also ran experiments to time the main primitives, i.e. switching and multiplication. We ran the multiply and switch functions 1000 times and took the mean time. Multiplication in the source group averages 0.0671s, while switching averages 0.0971s. The code is available at [REMOVED FOR ANONYMITY.]

	District	
	Albury (5 candidates)	**Auburn** (6 candidates)
No. Ballots	46347	43738
Ballot Encryption Time	3069s	3936s
No. Elimination Rounds	1	4
Count Time	6979s	54637s

Table 1. Results for Sample IRV Counts. Timings in seconds.

6 Conclusion

We have devised a very simple universally verifiable MPC protocol based on combining an efficient distributed key generation, a somewhat homomorphic cryptosystem in which one multiplication comes almost for free, and a switching protocol that allows a return to the cryptosystem from which more multiplications can be performed.

[4] For example, the use of a “stop” candidate by [30] remedies the case that a ballot is exhausted prematurely.

[5] From `http://pastvtr.elections.nsw.gov.au/SGE2015/la-home.htm`

Acknowledgement

Olivier Pereira is grateful to the Belgian Fund for Scientific Research (F.R.S.-FNRS) for its financial support provided through the the SeVoTe project, and to the Melbourne School of Engineering for its fellowship.

References

1. Abe, M., Fehr, S.: Perfect nizk with adaptive soundness. In: Vadhan, S.P. (ed.) Theory of Cryptography: 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007. Proceedings. pp. 118–136. Springer Berlin Heidelberg, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70936-7_7, https://doi.org/10.1007/978-3-540-70936-7_7
2. Adida, B., De Marneffe, O., Pereira, O., Quisquater, J.J.: Electing a university president using open-audit voting: Analysis of real-world use of helios. In: Proceedings of the 2009 Conference on Electronic Voting Technology/Workshop on Trustworthy Elections. pp. 10–10. EVT/WOTE'09, USENIX Association, Berkeley, CA, USA (2009), http://dl.acm.org/citation.cfm?id=1855491.1855501
3. Attrapadung, N., Hanaoka, G., Mitsunari, S., Sakai, Y., Shimizu, K., Teruya, T.: Efficient two-level homomorphic encryption in prime-order bilinear groups and a fast implementation in webassembly. In: Proceedings of the 2018 on Asia Conference on Computer and Communications Security. pp. 685–697. ACM (2018)
4. Barreto, P.S.L.M., Naehrig, M.: Pairing-friendly elliptic curves of prime order. In: Selected Areas in Cryptography – SAC'2005. LNCS, vol. 3897, pp. 319–331. Springer (2006)
5. Baum, C., Damgård, I., Orlandi, C.: Publicly auditable secure multi-party computation. In: International Conference on Security and Cryptography for Networks. pp. 175–196. Springer (2014), also Cryptology ePrint Archive, Report 2014/075: http://eprint.iacr.org/2014/075
6. Beimel, A.: Secure Schemes for Secret Sharing and Key Distribution. Ph.D. thesis, Israel Institute of Technology (1996)
7. Benaloh, J., Byrne, M., Kortum, P.T., McBurnett, N., Pereira, O., Stark, P.B., Wallach, D.S.: Star-vote: A secure, transparent, auditable, and reliable voting system. CoRR **abs/1211.1904** (2012), http://arxiv.org/abs/1211.1904
8. Benaloh, J., Moran, T., Naish, L., Ramchen, K., Teague, V.: Shuffle-sum: Coercion-resistant verifiable tallying for stv voting. Trans. Info. For. Sec. **4**(4), 685–698 (Dec 2009). https://doi.org/10.1109/TIFS.2009.2033757, http://dx.doi.org/10.1109/TIFS.2009.2033757
9. Bernhard, D., Cortier, V., Pereira, O., Smyth, B., Warinschi, B.: Adapting helios for provable ballot privacy. In: Vijay Atluri, C.D. (ed.) Computer Security – ESORICS 2011. Lecture Notes in Computer Science, Springer (9 2011)
10. Blum, M., Feldman, P., Micali, S.: Non-interactive zero-knowledge and its applications. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing. pp. 103–112. STOC '88, ACM, New York, NY, USA (1988). https://doi.org/10.1145/62212.62222, http://doi.acm.org/10.1145/62212.62222
11. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) Advances in Cryptology – CRYPTO 2004: 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19,

2004. Proceedings. pp. 41–55. Springer Berlin Heidelberg, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28628-8_3, https://doi.org/10.1007/978-3-540-28628-8_3

12. Boneh, D., Goh, E.J., Nissim, K.: Evaluating 2-dnf formulas on ciphertexts. In: Proceedings of the Second International Conference on Theory of Cryptography. pp. 325–341. TCC'05, Springer-Verlag, Berlin, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30576-7_18, http://dx.doi.org/10.1007/978-3-540-30576-7_18
13. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (leveled) fully homomorphic encryption without bootstrapping. ACM Trans. Comput. Theory **6**(3), 13:1–13:36 (Jul 2014)
14. Camenisch, J., Michels, M.: Separability and efficiency for generic group signature schemes. In: Wiener, M. (ed.) Advances in Cryptology — CRYPTO' 99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings. pp. 413–430. Springer Berlin Heidelberg, Berlin, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_27, https://doi.org/10.1007/3-540-48405-1_27
15. Castagnos, G., Imbert, L., Laguillaumie, F.: Encryption switching protocols revisited: Switching modulo p. In: Advances in Cryptology - CRYPTO 2017. LNCS, vol. 10401, pp. 255–287. Springer (2017)
16. Catalano, D., Fiore, D.: Using linearly-homomorphic encryption to evaluate degree-2 functions on encrypted data. In: Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security. pp. 1518–1529. CCS '15, ACM, New York, NY, USA (2015). https://doi.org/10.1145/2810103.2813624, http://doi.acm.org/10.1145/2810103.2813624
17. Cortier, V., Galindo, D., Glondu, S., Izabachène, M.: Distributed elgamal à la pedersen: Application to helios. In: Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society. pp. 131–142. WPES '13, ACM, New York, NY, USA (2013). https://doi.org/10.1145/2517840.2517852, http://doi.acm.org/10.1145/2517840.2517852
18. Couteau, G., Peters, T., Pointcheval, D.: Encryption switching protocols. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology – CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I. pp. 308–338. Springer Berlin Heidelberg, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_12, http://dx.doi.org/10.1007/978-3-662-53018-4_12
19. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Advances in Cryptology - EUROCRYPT '97. LNCS, vol. 1233, pp. 103–118. Springer (1997)
20. Damgard, I., Pastro, V., Smart, N., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. Cryptology ePrint Archive, Report 2011/535 (2011), http://eprint.iacr.org/2011/535
21. Damgård, I., Pastro, V., Smart, N., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Advances in Cryptology–CRYPTO 2012, pp. 643–662. Springer (2012)
22. Damgrd, I., Groth, J.: Non-interactive and reusable non-malleable commitment schemes. Cryptology ePrint Archive, Report 2003/080 (2003), http://eprint.iacr.org/2003/080
23. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust Non-interactive Zero Knowledge, pp. 566–598. Springer Berlin Heidelberg, Berlin,

Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_33, https://doi.org/10.1007/3-540-44647-8_33

24. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for diffie-hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II. pp. 129–147. Springer Berlin Heidelberg, Berlin, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_8, http://dx.doi.org/10.1007/978-3-642-40084-1_8
25. Freeman, D.M.: Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In: Proceedings of the 29th Annual International Conference on Theory and Applications of Cryptographic Techniques. pp. 44–61. EUROCRYPT'10, Springer-Verlag, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_3, http://dx.doi.org/10.1007/978-3-642-13190-5_3
26. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. J. Cryptol. **20**(1), 51–83 (Jan 2007). https://doi.org/10.1007/s00145-006-0347-3, http://dx.doi.org/10.1007/s00145-006-0347-3
27. Goh, E.J., Golle, P.: Event driven private counters. In: Proceedings of the 9th International Conference on Financial Cryptography and Data Security. pp. 313–327. FC'05, Springer-Verlag, Berlin, Heidelberg (2005). https://doi.org/10.1007/11507840_27, http://dx.doi.org/10.1007/11507840_27
28. Groth, J.: Short non-interactive zero-knowledge proofs. In: Abe, M. (ed.) Advances in Cryptology - ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings. pp. 341–358. Springer Berlin Heidelberg, Berlin, Heidelberg (2010). https://doi.org/10.1007/978-3-642-17373-8_20, http://dx.doi.org/10.1007/978-3-642-17373-8_20
29. Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive zaps and new techniques for nizk. In: Dwork, C. (ed.) Advances in Cryptology - CRYPTO 2006: 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006. Proceedings. pp. 97–111. Springer Berlin Heidelberg, Berlin, Heidelberg (2006). https://doi.org/10.1007/11818175_6, https://doi.org/10.1007/11818175_6
30. Heather, J.: Implementing stv securely in prêt à voter. In: Proceedings of the 20th IEEE Computer Security Foundations Symposium. pp. 157–169. CSF '07, IEEE Computer Society, Washington, DC, USA (2007). https://doi.org/10.1109/CSF.2007.22, http://dx.doi.org/10.1109/CSF.2007.22
31. Herold, G., Hesse, J., Hofheinz, D., Ràfols, C., Rupp, A.: Polynomial spaces: A new framework for composite-to-prime-order transformations. Cryptology ePrint Archive, Report 2014/445 (2014), http://eprint.iacr.org/2014/445
32. Ito, M., Saito, A., Nishizeki, T.: Secret sharing scheme realizing general access structure. Electronics and Communications in Japan (Part III: Fundamental Electronic Science) **72**(9), 56–64 (1989). https://doi.org/10.1002/ecjc.4430720906, https://onlinelibrary.wiley.com/doi/abs/10.1002/ecjc.4430720906
33. Park, C., Itoh, K., Kurosawa, K.: Efficient anonymous channel and all/nothing election scheme. In: Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology. pp. 248–259. EUROCRYPT '93, Springer-

Verlag New York, Inc., Secaucus, NJ, USA (1994), `http://dl.acm.org/citation.cfm?id=188307.188351`
34. Pedersen, T.P.: A threshold cryptosystem without a trusted party. In: Proceedings of the 10th Annual International Conference on Theory and Application of Cryptographic Techniques. pp. 522–526. EUROCRYPT'91, Springer-Verlag, Berlin, Heidelberg (1991), `http://dl.acm.org/citation.cfm?id=1754868.1754929`
35. Ryan, P.Y.A.: Prêt à voter with paillier encryption. Math. Comput. Model. **48**(9-10), 1646–1662 (Nov 2008). https://doi.org/10.1016/j.mcm.2008.05.015, `http://dx.doi.org/10.1016/j.mcm.2008.05.015`
36. Ryan, P.Y.A.: A variant of the chaum voter-verifiable scheme. In: Proceedings of the 2005 Workshop on Issues in the Theory of Security. pp. 81–88. WITS '05, ACM, New York, NY, USA (2005). https://doi.org/10.1145/1045405.1045414, `http://doi.acm.org/10.1145/1045405.1045414`
37. Ryan, P.Y.A., Teague, V.: Ballot permutations in prêt à voter. In: Proceedings of the 2009 Conference on Electronic Voting Technology/Workshop on Trustworthy Elections. pp. 13–13. EVT/WOTE'09, USENIX Association, Berkeley, CA, USA (2009), `http://dl.acm.org/citation.cfm?id=1855491.1855504`
38. Schoenmakers, B., Tuyls, P.: Practical two-party computation based on the conditional gate. In: Lee, P.J. (ed.) Advances in Cryptology - ASIACRYPT 2004: 10th International Conference on the Theory and Application of Cryptology and Information Security, Jeju Island, Korea, December 5-9, 2004. Proceedings. pp. 119–136. Springer Berlin Heidelberg, Berlin, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30539-2_10, `https://doi.org/10.1007/978-3-540-30539-2\_10`
39. Schoenmakers, B., Veeningen, M.: Universally verifiable multiparty computation from threshold homomorphic cryptosystems. In: International Conference on Applied Cryptography and Network Security. pp. 3–22. Springer (2015), cryptology ePrint Archive, 2015/058: `http://eprint.iacr.org/2015/058`
40. Scott, M.: Authenticated id-based key exchange and remote log-in with simple token and pin number. Cryptology ePrint Archive, Report 2002/164 (2002), `http://eprint.iacr.org/2002/164`
41. Waters, B.: Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In: Proceedings of the 14th International Conference on Practice and Theory in Public Key Cryptography Conference on Public Key Cryptography. pp. 53–70. PKC'11, Springer-Verlag, Berlin, Heidelberg (2011), `http://dl.acm.org/citation.cfm?id=1964658.1964664`

A Reusable Commitments

Definition 9 (Commitment Scheme [22]) *A commitment scheme* π_{COM} *consists of three PPT algorithms* $(\mathcal{K}, \mathsf{commit}_{ck}, \mathsf{decommit}_{ck})$.

$\mathcal{K}(1^\lambda)$**:** *On input* 1^λ*, the key generator outputs a public key ck. Associated to this are a message space* $\mathcal{M}_{ck}$*, a commitment space* $\mathcal{C}_{ck}$ *and two polynomial time algorithms* commit_{ck} *and* $\mathsf{decommit}_{ck}$.

$\mathsf{commit}_{ck}(m, r)$ *: On input* $m \in \mathcal{M}_{ck}$ *choose at random a randomiser r. The output is* (c, d) *where* $c \in \mathcal{C}_{ck}$ *and d is some decommitment information.*

$\mathsf{decommit}_{ck}(c, d)$ *: If* $c \notin \mathcal{C}_{ck}$ *or d is not a proper opening output* $\perp$*. Otherwise, if c is constructed as the output of* commit_{ck} *output m.*

Definition 10 (Non-malleable Commitment Scheme [22]) *Let $\mathcal{K}'$ be a modified key generator which outputs a public key indistinguishable from the real key. Let $\mathcal{A}$ be a PPT adversary. Let M be a message generator and D be distinguisher which receive as auxiliary input z_M and z_D respectively. We require that for every such adversary $\mathcal{A}$ there exists a PPT simulator $\mathcal{S}$ so that following ensembles are computationally indistinguishable.*

$$\begin{gathered}
\{D(s, \boldsymbol{m}, \boldsymbol{m}', z_D) : ck \leftarrow \mathcal{K}(1^\lambda), (s, \boldsymbol{m}) \leftarrow M(ck, z_M); \\
(\boldsymbol{c}, \boldsymbol{d}) \leftarrow \mathsf{commit}_{ck}(\boldsymbol{m}); \boldsymbol{c}' \leftarrow \mathcal{A}(ck, \boldsymbol{c}, M, z_M); \boldsymbol{d}' \leftarrow \mathcal{A}(\boldsymbol{d}); \\
\boldsymbol{m}' \leftarrow \mathsf{decommit}_{ck}(\boldsymbol{c}', \boldsymbol{d}')\}_{\lambda, z_M, z_D} \\
\approx_c \\
\{D(s, \boldsymbol{m}, \boldsymbol{m}', z_D) : (ck, s_{ck}) \leftarrow \mathcal{K}'(1^\lambda); (s, \boldsymbol{m}) \leftarrow M(ck, z_M); \\
\boldsymbol{m}' \leftarrow \mathcal{S}(ck, s_{ck}, |\boldsymbol{m}|, M, z_M)\}_{\lambda, z_M, z_D}
\end{gathered}$$

B Encryption Switching

In this section we define an N-party extension of the encryption switching protocols by Couteau *et al.* [18] as well as a definition of security following the simulation-based paradigm introduced there-in. We will need the notion of twin-ciphertext pair [18] which is augmented with appropriate homomorphic properties on the respective ciphertext spaces.

Definition 11 (Twin-Ciphertext Pair [18]) *For $i = \{1, 2\}$ let Π_i be an encryption scheme $(\mathsf{Setup}_i, \mathsf{KeyGen}_i, \mathsf{Enc}_i, \mathsf{Dec}_i)$ with plaintext space $\mathcal{M}_i$. A twin-ciphertext pair (c_1, c_2) is a pair of ciphertexts satisfying:*

1. *c_1 is an encryption of $m_1 \in \mathcal{M}_1$ under Π_1.*
2. *c_2 is an encryption of $m_2 \in \mathcal{M}_2$ under Π_2.*
3. *$m_1 = m_2$ (which in turn belongs to $\mathcal{M}_1 \cap \mathcal{M}_2$).*

Definition 12 (Distributed Encryption Switching) *For $i \in \{1, 2\}$ let $\Pi_i = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_i, \mathsf{Dec}_i)$ be semantically secure cryptosystems with plaintext spaces $(\mathcal{M}_i, +, \times)$ such that $\mathcal{M} = \mathcal{M}_1 \cap \mathcal{M}_2 \neq \emptyset$. Suppose that the following homomorphic properties hold:*

$$\begin{aligned}
&\forall m, m_1, m_2 \in \mathcal{M}_i \\
&\mathsf{Enc}_i(m_1; r_1) \cdot_i \mathsf{Enc}_i(m_2; r_2) = \mathsf{Enc}_i(m_1 + m_2; r_1 + r_2), \\
&\forall R \in \mathcal{M} \quad (\mathsf{Enc}_i(m; r))^R = \mathsf{Enc}_i(R \times m; R \cdot r)
\end{aligned}$$

Assume also the existence of $\mathsf{Rand}_i(\cdot)$ which re-randomizes a ciphertext in Π_i (for example, these can be constructed via multiplication of the input ciphertext with a fresh encryption of zero using the above homomorphisms).

A N-party distributed encryption switching protocol between Π_1 and Π_2, with respect to access structure $\mathbb{A}$, denoted $\Pi_1 \rightleftharpoons \Pi_2$, is a tuple $(\mathsf{Share}, \mathsf{Switch})$ such that:

$\mathsf{Share}(\mathsf{pk}, \mathsf{sk}, \mathbb{A})$ *Given input* sk *outputs a secret sharing* $(\mathsf{sk}_1, \ldots, \mathsf{sk}_N)$*, according to access structure* $\mathbb{A}$ *and updates* pk *if necessary.*

$\mathsf{Switch}_{\mathsf{par}}(\mathsf{pk}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_N), c)$ *is an interactive protocol which from a ciphertext* c *under the source encryption scheme, jointly computes a twin ciphertext* c' *of* c *under the destination encryption scheme or outputs* $\perp$ *(in case of incorrect execution of the protocol). Here subscript* par *indicates the direction of the encryption switching.*

Definition 13 (Correctness of Distributed Encryption Switching) *A distributed encryption switching scheme* $\Pi_1 \rightleftharpoons \Pi_2 = (\mathsf{Share}, \mathsf{Switch})$ *is correct if both* Π_1 *and* Π_2 *are correct encryption schemes, and for any* $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\kappa)$*, any keys* $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(\mathsf{pp})$*, any key shares* pk *and* $(\mathsf{sk}_1, \ldots, , \mathsf{sk}_N) \leftarrow \mathsf{Share}(\mathsf{pk}, \mathsf{sk})$*, any message* $m \in \mathcal{M}_1 \cap \mathcal{M}_2$ *and any* $c_i \leftarrow \mathsf{Enc}_i(\mathsf{pk}_i, m)$ *for* $i = 1, 2$*,*

$$\mathsf{Dec}_2(\mathsf{sk}, \mathsf{Switch}_{1\to 2}(\mathsf{pk}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_N), c_1)) = m,$$
$$\mathsf{Dec}_1(\mathsf{sk}, \mathsf{Switch}_{2\to 1}(\mathsf{pk}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_N), c_2)) = m$$

Definition 14 (Security of Distributed Encryption Switching) *Figure 6 shows two experiments in which an adversary interacts with an* N*-party encryption switching scheme over access structure* $\mathbb{A}$*. In the first experiment the adversary interacts with the real encryption switching protocol on input ciphertext* c*. In the second experiment the adversary interacts with a simulator that is given input* $(c, \overline{c})$*, which is a twin-ciphertext pair. Let* $\mathcal{O}_V$ *be an oracle which on input* $(\mathsf{pk}, (\mathsf{sk}_i)_{P_i \in V}, c)$*, emulates the honest players in set* $V \subseteq \{P_1, \ldots, P_N\}$*, i.e., provides the answers* P_i *would send upon receiving* Start *when running the protocol* $\mathsf{Switch}(\mathsf{pk}, (\mathsf{sk}_1, \ldots, \mathsf{sk}_N), c)$*, for each* P_i *in* V*.*

```
Experiment $\mathsf{RealSwitch}_{\mathbb{A}}^{\mathsf{desp},\ \mathcal{A}}(1^\lambda, 1^N)$ :
    $B \leftarrow \mathcal{A}(1^\lambda, 1^N) : B \notin \mathbb{A}$
    $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$
    $(\mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathsf{Share}(\mathsf{pk}, \mathsf{sk})$
    $\alpha \leftarrow \mathcal{A}^{\mathcal{O}_{\overline{B}}(\mathsf{pk}, (\mathsf{sk}_i)_{i \in \overline{B}}, c)}(\mathsf{pk}, (\mathsf{sk}_i)_{i \in B})$
    Output: $\alpha$

Experiment $\mathsf{IdealSwitch}_{\mathbb{A},\ (\mathcal{S}_1, \mathcal{S}_2)}^{\mathsf{desp},\ \mathcal{A}}(1^\lambda, 1^N)$ :
    $B \leftarrow \mathcal{A}(1^\lambda, 1^N) : B \notin \mathbb{A}$
    $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$
    $(\mathsf{pk}', \mathsf{sk}'_1, \ldots, \mathsf{sk}'_N) \leftarrow \mathcal{S}_1(\mathsf{pk})$
    $\alpha \leftarrow \mathcal{A}^{\mathcal{S}_2(\mathsf{pk}, (\mathsf{sk}'_i)_{i \in \overline{B}}, c, \overline{c})}(\mathsf{pk}', (\mathsf{sk}'_i)_{i \in B})$
    Output: $\alpha$
```

Fig. 6. Experiments used in distributed encryption switching.

An N*-party distributed encryption switching scheme* $\Pi_1 \rightleftharpoons \Pi_2$ *is* $(N, \mathbb{A})$*-simulation secure, if for every PPT adversary* $\mathcal{A}$ *there exists a PPT simulator* $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ *such that the ensembles* $\{\mathsf{RealSwitch}_{\mathbb{A}}^{\mathsf{desp},\ \mathcal{A}}(1^\lambda, 1^N)\}_{\lambda, N}$ *and* $\{\mathsf{IdealSwitch}_{\mathbb{A},\ (\mathcal{S}_1, \mathcal{S}_2)}^{\mathsf{desp},\ \mathcal{A}}(1^\lambda, 1^N)\}_{\lambda, N}$ *are computationally indistinguishable.*

C One-time Multiplicatively Homomorphic Cryptosystem

In this section we describe a generalisation of the homomorphic cryptosystem from Section 2 which supports arbitrarily many additions on the message space, followed by one multiplication, followed by arbitrarily many additions. The purpose of this appendix is to detail the case for general l, from which the cryptosystem in Section 2 may be seen as the special case $l = 1$.

We first present the general definition of the tensor-induced projecting bilinear generator.

Definition 15 (Tensor-Induced Projecting Bilinear Group Generator) *Let $\mathcal{P}$ be prime-order bilinear group generator and let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \hat{e}) \leftarrow \mathcal{P}(1^\lambda)$. Let $l \in \mathbb{N}$ be a constant and $G = \mathbb{G}_1^{l+1}, H = \mathbb{G}_2^{l+1}$ and $G_t = \mathbb{G}_t^{(l+1)^2}$. Choose generators $g \in_R \mathbb{G}_1, h \in_R \mathbb{G}_2$. In what follows define $R(\boldsymbol{y}, i)$ to be the vector $\boldsymbol{y}$ cyclically shifted right by i positions. Let $\boldsymbol{a} \otimes \boldsymbol{b}$ be the tensor product of $\boldsymbol{a}$ and $\boldsymbol{b}$.*

1. *Let $\boldsymbol{x}_1 = (-s, 1, 0, \dots, 0)$,*
   $\boldsymbol{x}_2 = R(\boldsymbol{x}_1, 1), \dots,$
   $\boldsymbol{x}_l = R(\boldsymbol{x}_1, l-1)$ and
   $\boldsymbol{x}'_1 = (-s', 1, 0, \dots, 0)$,
   $\boldsymbol{x}'_2 = R(\boldsymbol{x}'_1, 1), \dots,$
   $\boldsymbol{x}'_l = R(\boldsymbol{x}'_1, l-1)$ in $\mathbb{Z}_p^{l+1}$ and
   $\boldsymbol{y}_{\boldsymbol{1}} = \boldsymbol{x}_1 \otimes \boldsymbol{x}'_1, \dots,$
   $\boldsymbol{y}_{l(i-1)+j} = \boldsymbol{x}_i \otimes \boldsymbol{x}'_j, \dots,$
   $\boldsymbol{y}_{\boldsymbol{l^2}} = \boldsymbol{x}_l \otimes \boldsymbol{x}'_l$ in $\mathbb{Z}_p^{(l+1)^2}$ where s and $s' \in_R \mathbb{Z}_p$.
2. *Let G_1 be the subgroup of G generated by $\{g^{\boldsymbol{x}_1}, \dots, g^{\boldsymbol{x}_l}\}$ and H_1 be the subgroup of H generated by $\{h^{\boldsymbol{x}'_1}, \dots, h^{\boldsymbol{x}'_l}\}$. Let G'_t be the subgroup of G_t generated by $\{\hat{e}(g,h)^{\boldsymbol{y}_1}, \dots, \hat{e}(g,h)^{\boldsymbol{y}_{l^2}}\}$.*
3. *Define $e : G \times H \to G_t$ by*
$$\begin{aligned} e(g^{\boldsymbol{a}}, h^{\boldsymbol{b}}) &=: \hat{e}(g,h)^{\boldsymbol{a} \otimes \boldsymbol{b}} \\ &= (\hat{e}(g^{a_0}, h^{b_0}), \dots, \\ &\qquad \hat{e}(g^{a_i}, h^{b_j}), \dots, \\ &\qquad\qquad \hat{e}(g^{a_l}, h^{b_l})) \end{aligned}$$
   where $\boldsymbol{a} = (a_0, \dots, a_l)$ and $\boldsymbol{b} = (b_0, \dots, b_l)$.
4. *For $\mathbf{g} \in G, \mathbf{h} \in H$ and $\mathbf{g}_t \in G_t$ and $s, s' \in \mathbb{Z}_p$, define*
$$\pi_1(\mathbf{g}) = \mathbf{g}^{(1,s,\dots,s^l)^T} \quad \pi_2(\mathbf{h}) = \mathbf{h}^{(1,s',\dots,s'^l)^T}$$
$$\pi_t(\mathbf{g}_t) = \mathbf{g}_t^{((s^i s'^j)_{i,j=0}^l)^T}$$
   which are elements in $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_t$ respectively.
5. *Output $(G, G_1, H, H_1, G_t, G'_t, e)$ and (π_1, π_2, π_t).*

Using this construction, the cryptosystem is designed as follows.

$\mathsf{Setup}(1^\lambda)$: Let $\mathcal{P}$ be the prime-order bilinear generator of Definition 8. Output $\mathsf{pp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \hat{e}) \leftarrow \mathcal{P}(1^\lambda)$.

$\mathsf{KeyGen}(\mathsf{pp})$: Let $\mathcal{G}$ be the tensor-induced projecting bilinear generator of Definition 15. Let $(G, G_1, H, H_1, G_t, G'_t, e, \pi_1, \pi_2, \pi_t) \leftarrow \mathcal{G}(\mathsf{pp})$. In particular, let $\mathbb{G}_1^{l+1}, \mathbb{G}_2^{l+1}$ and $\mathbb{G}_t^{(l+1)^2}$ be descriptions of G, H and G_t respectively, and $\{g^{\boldsymbol{x}_1}, \ldots, g^{\boldsymbol{x}_l}\}$ and $\{h^{\boldsymbol{x}'_l}, \ldots, h^{\boldsymbol{x}'_l}\}$ be descriptions of G_1, H_1 respectively. Choose $\mathbf{g} \in_R G$, $\mathbf{h} \in_R H$, and output the public key $\mathsf{pk} = (G, G_1, H, H_1, G_t, e, \mathbf{g}, \mathbf{h})$ and the secret key $\mathsf{sk} = (\pi_1, \pi_2, \pi_t)$ as described in Section 1.5.

$\mathsf{Enc_{src}}(\mathsf{pk}, M)$: Choose $(a_i)_{1\le i\le l}$ and $(b_i)_{1\le i\le l}$ at random in $\mathbb{Z}_p$. Let $\mathbf{g}_1 = \prod_{j=1}^{l} (g^{\boldsymbol{x}_j})^{a_j} = (g^{-a_1 s}, g^{a_1 - a_2 s}, \ldots, g^{a_{l-1} - a_l s}, g^{a_l})$ and $\mathbf{h}_1 = \prod_{j=1}^{l} (h^{\boldsymbol{x}'_j})^{b_j} = (h^{-b_1 s'}, h^{b_1 - b_2 s'}, \ldots, h^{b_{l-1} - b_l s'}, h^{b_l})$. Let $C_0 = \mathbf{g}^M \cdot \mathbf{g}_1, C_1 = \mathbf{h}^M \cdot \mathbf{h}_1$. Output the ciphertext (C_0, C_1) in $G \times H$.

$\mathsf{Enc_{tgt}}(\mathsf{pk}, M)$: Choose $(a_i)_{1\le i\le l}$ and $(b_i)_{1\le i\le l}$ at random in $\mathbb{Z}_p$. Let $\mathbf{g}_1 = \prod_{j=1}^{l} (g^{\boldsymbol{x}_j})^{a_j} = (g^{-a_1 s}, g^{a_1 - a_2 s}, \ldots, g^{a_{l-1} - a_l s}, g^{a_l})$ and $\mathbf{h}_1 = \prod_{j=1}^{l} (h^{\boldsymbol{x}'_j})^{b_j} = (h^{-b_1 s'}, h^{b_1 - b_2 s'}, \ldots, h^{b_{l-1} - b_l s'}, h^{b_l})$. Output the ciphertext $C = e(\mathbf{g}, \mathbf{h})^M \cdot e(\mathbf{g}, \mathbf{h}_1) \cdot e(\mathbf{g}_1, \mathbf{h})$ in G_t.

$\mathsf{Multiply_{src}}(\mathsf{pk}, C, C')$: The multiplication algorithm takes as input two ciphertexts $C = (C_0, C_1)$ and $C' = (C'_0, C'_1)$. Choose $\mathbf{g}_1 \in_R G_1$ and $\mathbf{h}_1 \in_R H_1$, as in the above routine. Output $C = e(C_0, C'_1) \cdot e(\mathbf{g}, \mathbf{h}_1) \cdot e(\mathbf{g}_1, \mathbf{h})$, an element of G_t.

$\mathsf{Add_{src}}(\mathsf{pk}, C, C')$: The algorithm accepts as inputs two ciphertexts $C = (C_0, C_1)$ and $C' = (C'_0, C'_1)$. Choose $\mathbf{g}_1 \in_R G_1$ and $\mathbf{h}_1 \in_R H_1$.
1. Let $C''_0 = C_0 \cdot C'_0 \cdot \mathbf{g}_1$.
2. Let $C''_1 = C_1 \cdot C'_1 \cdot \mathbf{h}_1$.

Output $C'' = (C''_0, C''_1)$.

$\mathsf{Add_{tgt}}(\mathsf{pk}, C, C')$: The algorithm accepts as inputs two ciphertexts C and C' in G_t. Choose $\mathbf{g}_1 \in_R G_1$ and $\mathbf{h}_1 \in_R H_1$.
1. Let $C'' = C \cdot C' \cdot e(\mathbf{g}, \mathbf{h}_1) \cdot e(\mathbf{g}_1, \mathbf{h})$.

Output C''.

$\mathsf{Dec_{src}}(\mathsf{sk}, C)$: Accept as input a ciphertext $C = (C_0, C_1)$ in $G \times H$. Compute $M \leftarrow \log_{\pi_1(\mathbf{g})}(\pi_1(C_0))$ and $M' \leftarrow \log_{\pi_2(\mathbf{h})}(\pi_2(C_1))$. Output M if $M = M'$ or $\perp$ otherwise.

$\mathsf{Dec_{tgt}}(\mathsf{sk}, C)$: Accept as input a ciphertext C in G_t. Output $M \leftarrow \log_{\pi_t(e(\mathbf{g},\mathbf{h}))}(\pi_t(C))$.

Proof of Lemma 1 Suppose that the External l-Symmetric Cascade assumption holds with respect to the groups $\mathbb{G}_1$ and $\mathbb{G}_2$. Then the above cryptosystem is semantically secure.

Proof. We prove this via a series of games, of which the indistinguishability is proven in the next 3 Propositions.

Game H_1: Exactly the same as above but modify the encryption routine as follows.

$\mathsf{Encrypt}(\mathsf{pk}, M)$: Choose $(u_i)_{1\leq i\leq l+1}$ and $(b_i)_{1\leq i\leq l+1}$ at random in $\mathbb{Z}_p$. Let $C_0 = \mathbf{g}^M \cdot (g^{u_1}, \ldots, g^{u_{l+1}}), C_1 = \mathbf{h}^M \cdot (h^{-b_1 s'}, h^{b_1 - b_2 s'}, h^{b_{l-1} - b_l s'}, h^{b_l})$. Output the ciphertext (C_0, C_1) in $G \times H$.

Game H_2**:** Exactly the same as $\mathbf{H_1}$ but modify the encryption routine as follows.

$\mathsf{Encrypt}(\mathsf{pk}, M)$: Choose $(u_i)_{1\leq i\leq l+1}$ and $(u'_i)_{1\leq i\leq l+1}$ at random in $\mathbb{Z}_p$. Let $C_0 = \mathbf{g}^M \cdot (g^{u_1}, \ldots, g^{u_{l+1}}), C_1 = \mathbf{h}^M \cdot (h^{u'_1}, \ldots, h^{u'_{l+1}})$. Output the ciphertext (C_0, C_1) in $G \times H$.

Proposition 6 *Suppose there exists a PPT adversary $\mathcal{A}$ that distinguishes the real IND-CPA game and H_1 with probability ϵ_1. Then we can construct a PPT adversary $\mathcal{B}$ that breaks the l-Symmetric Cascade assumption in $\mathbb{G}_1$ with advantage ϵ_1.*

Proof. On input $(\mathbb{G}, g, g^A, g^{\boldsymbol{v}})$, attacker $\mathcal{B}$ performs the following steps. Write $g^A = g^{\boldsymbol{x}_1} \| \ldots \| g^{\boldsymbol{x}_l}$. Set $G_1 = \{g^{\boldsymbol{x}_1}, \ldots, g^{\boldsymbol{x}_l}\}$. Send pk to $\mathcal{A}$. On receipt of (M_0, M_1), choose $\beta \in_R \{0,1\}$. Let $CT = (C_1, C_2)$, where $C_0 = \mathbf{g}^{M_\beta} \cdot g^{\boldsymbol{v}}, C_1 = \mathbf{h}^{M_\beta} \cdot \mathbf{h}_1 : \mathbf{h}_1 \in_r H_1$. Send CT to $\mathcal{A}$. Output the bit that $\mathcal{A}$ outputs. Clearly $\Pr[1 \leftarrow \mathcal{B}(\mathbb{G}_1, g, (g^{\boldsymbol{x}_1}, \ldots, g^{\boldsymbol{x}_l}), g^{\sum_{j=1}^{l} a_j \boldsymbol{x}_j})] = \Pr[1 \leftarrow \mathcal{A}(H_0)]$, while $\Pr[1 \leftarrow \mathcal{B}(\mathbb{G}_1, g, (g^{\boldsymbol{x}_1}, \ldots, g^{\boldsymbol{x}_l}), g^{(u_1, \ldots, u_{l+1})^T})] = \Pr[1 \leftarrow \mathcal{A}(H_1)]$. Thus $Adv_{\mathcal{B}} = |\Pr[1 \leftarrow \mathcal{A}(H_0)] - \Pr[1 \leftarrow \mathcal{A}(H_1)]| = Adv_{\mathcal{A}} = \epsilon_1$.

Proposition 7 *Suppose there exists a PPT adversary $\mathcal{A}$ that distinguishes H_1 and H_2 with probability ϵ_2. Then we can construct a PPT adversary $\mathcal{B}$ that breaks the l-Symmetric Cascade assumption in $\mathbb{G}_2$ with advantage ϵ_2.*

Proof. On input $(\mathbb{G}_2, h, h^A, h^{\boldsymbol{v}'})$, attacker $\mathcal{B}$ performs the following steps. Write $h^A = h^{\boldsymbol{x}'_1} \| \ldots \| h^{\boldsymbol{x}'_l}$. Set $H_1 = \{h^{\boldsymbol{x}'_1}, \ldots, h^{\boldsymbol{x}'_l}\}$. Send pk to $\mathcal{A}$. On receipt of (M_0, M_1), choose $\beta \in_R \{0,1\}$. Let $CT = (C_1, C_2)$, where $C_0 = \mathbf{g}^{M_\beta} \cdot (g^{u_1}, \ldots, g^{u_{l+1}}) : u_1, \ldots, u_{l+1} \in_R \mathbb{Z}_p, C_1 = \mathbf{h}^{M_\beta} \cdot h^{\boldsymbol{v}'}$. Send CT to $\mathcal{A}$. Output the bit that $\mathcal{A}$ outputs. Clearly $\Pr[1 \leftarrow \mathcal{B}(\mathbb{G}_2, h, (h^{\boldsymbol{x}'_1}, \ldots, h^{\boldsymbol{x}'_l}), h^{\sum_{j=1}^{l} b_j \boldsymbol{x}'_j})] = \Pr[1 \leftarrow \mathcal{A}(H_1)]$, while $\Pr[1 \leftarrow \mathcal{B}(\mathbb{G}_2, h, (h^{\boldsymbol{x}'_1}, \ldots, h^{\boldsymbol{x}'_l}), h^{(u'_1, \ldots, u'_{l+1})^T})] = \Pr[1 \leftarrow \mathcal{A}(H_2)]$. Thus $Adv_{\mathcal{B}} = |\Pr[1 \leftarrow \mathcal{A}(H_1)] - \Pr[1 \leftarrow \mathcal{A}(H_2)]| = Adv_{\mathcal{A}} = \epsilon_2$.

Proposition 1. *Any PPT adversary $\mathcal{A}$ has negligible advantage in winning the modified IND-CPA game H_2.*

Proof. This follows from the fact that the challenge $CT = (C_0, C_1)$ carries no information about the challenge M_β.

Combining the above propositions, we have that any IND-CPA adversary has advantage at most $\epsilon_1 + \epsilon_2$ against the above cryptosystem. Therefore if the External l-Symmetric Cascade assumption holds, ϵ_1 and ϵ_2 are negligible, thus semantic security of the cryptosystem follows.

D NIZKs

In this section we present non-interactive zero knowledge proofs for the relations described in Section 3. For convenience our presentation of these is unified - our proofs assume inputs under the verifiable commitment scheme of [29] while known techniques can be used to interchange between commitments under this scheme and ciphertexts under the other cryptosystems in this paper [14].

Plaintext equality proofs are denoted $P_{\mathsf{eq}}(C_1, C_2)$. We describe an adaption of the proof by Abe and Fehr [1] to produce a NIZK proof of equality of encrypted messages under different public keys. Proof $P_{\mathsf{range}}(C, 2^\lambda)$ uses standard bit decomposition of the input to prove that the committed value is in the range $[0, 2^\lambda - 1]$. We use a version adapted to the decision linear commitments setting. These in turn rely on the homomorphic commitment scheme of [29], which is secure if the Decision Linear assumption holds.

Our proofs are adapted for the Decision Linear Assumption.

Definition 16 (Decision Linear Assumption [11]) *Let g, u, v, h be generators in $\mathbb{G}$. For adversary $\mathcal{A}$ define*

$$\begin{aligned}&\mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{A}} := \\ &|\Pr[\mathcal{A}(u, v, h, u^a, v^b, h^{a+b}) = \mathsf{true} : u, v, h \in_R \mathbb{G}, a, b \in_R \mathbb{Z}_p] - \\ &\ \Pr[\mathcal{A}(u, v, h, u^a, v^b, y) = \mathsf{true} : u, v, h, y \in_R \mathbb{G}, a, b \in_R \mathbb{Z}_p]|\end{aligned}$$

Then for all PPT adversaries $\mathcal{A}$ we have $\mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{A}}$ is a negligible function of λ.

D.1 Non-Interactive Zero Knowledge

We recall the notion of a non-interactive zero knowledge proof system [10]. Standard techniques can be used to make such proofs non-malleable where necessary [23].

Definition 17 (Non-Interactive Zero Knowledge Proof [28]) *A non-interactive zero knowledge proof system for a relation $\mathcal{R}$ is a tuple (G, P, V) such that*

$G(1^\lambda, m)$**:** *a common reference string generator that takes as input the security parameter written in unary and an intended statement size m and outputs a common reference string σ of length $\Omega(\lambda)$.*

$P(\sigma, x, w)$**:** *a prover algorithm that takes as input the common reference string σ, statement x and witness w such that $\mathcal{R}(x, w)$ and outputs a proof ε.*

$V(\sigma, x, \varepsilon)$**:** *the verifier algorithm that on input the common reference string σ, the statement x and claimed proof, ε, outputs 1 or 0, indicating acceptance or rejection respectively.*

Additionally the following properties should hold:

Completeness. *For all PPT adversaries $\mathcal{A}$ and $m < \lambda^c$ for some $c > 0$ we have*
$\Pr[\sigma \leftarrow G(1^\lambda, m); (x, w) \leftarrow \mathcal{A}(\sigma), \varepsilon \leftarrow P(\sigma, x, w) : \mathcal{R}(x, w) \Rightarrow V(\sigma, x, \varepsilon) = 1] = 1$

Soundness. *For all PPT adversaries $\mathcal{A}$ and $m < \lambda^c$ for some $c > 0$ we have* $\Pr[\sigma \leftarrow G(1^\lambda, m); (x, \varepsilon) \leftarrow \mathcal{A}(\sigma) : x \notin L_m \wedge V(\sigma, x, \varepsilon) = 1] \approx 0$

Computational Zero Knowledge. *For all non-uniform polynomial time stateful adversaries $\mathcal{A}$, i.e., adversaries which accepts an advice string dependent on the input length, there exists a polynomial time simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ such that*

$$\begin{aligned}
&\Pr[\sigma \leftarrow G(1^\lambda, m); (x, w) \leftarrow \mathcal{A}(\sigma); \varepsilon \leftarrow P(\sigma, x, w) :\\
&(x, w) \in \mathcal{R}_m \wedge \mathcal{A}(\varepsilon) = 1]\\
&\approx_c\\
&\Pr[(\sigma, \tau) \leftarrow \mathcal{S}_1(1^\lambda, m); (x, w) \leftarrow \mathcal{A}(\sigma); \varepsilon \leftarrow \mathcal{S}_2(\tau, x) :\\
&(x, w) \in \mathcal{R}_m \wedge \mathcal{A}(\varepsilon) = 1]
\end{aligned}$$

Decision Linear Commitments [29] We present the homomorphic commitment scheme of [29] which is secure if the Decision Linear assumption holds. We require a slight twist on this scheme, which is that parameters will consist of *pairs* of elements from a group $G = \mathbb{G}^m$, $m \geq 3$, for which a symmetric bilinear map $e : G \times G \rightarrow G_t$ exists. The reason for this modification will become apparent when present the NIZK proof systems themselves. All operations are performed component-wise unless otherwise specified.

Setup :
Let $(p, G, e, g) \leftarrow \mathcal{G}(1^\lambda)$. Let $\mathbf{g} \in_R G^2$. Let $x, y \leftarrow \mathbb{Z}_p^{2m}$. Let $\mathbf{f} = \mathbf{g}^x, \mathbf{h} = \mathbf{g}^y$. Let $\mathsf{pk} = (p, \mathbb{G}, \mathbb{G}_T, e, \mathbf{g}, \mathbf{f}, \mathbf{h})$. Let $\mathsf{sk} = (\mathsf{pk}, x, y)$.

Perfectly hiding key generation K_{hide} :
1. $\mathbf{u}, \mathbf{v} \in_R G^2$. Let $\mathbf{w} = \mathbf{u}^{x^{-1}} \mathbf{v}^{y^{-1}}$.
2. Return $ck = (\mathsf{pk}, \mathbf{u}, \mathbf{v}, \mathbf{w})$.

Perfectly binding key generation K_{bind} :
1. $\mathbf{u}, \mathbf{v}, \mathbf{w} \in_R G^2$.
2. Return $ck = (\mathsf{pk}, \mathbf{u}, \mathbf{v}, \mathbf{w})$.

Commitment :
To commit to message $\mu \in \mathbb{Z}_p$ do
1. $r, s \leftarrow \mathbb{Z}_p^{2m}$
2. Return $c = (c_1, c_2, c_3) = \mathsf{com}(\mu; r, s) = (\mathbf{u}^\mu \mathbf{f}^r, \mathbf{v}^\mu \mathbf{h}^s, \mathbf{w}^\mu \mathbf{g}^{r+s})$.

Trapdoor opening :
Given a commitment $c = \mathsf{com}(\mu; r, s)$ under a perfectly hiding key, we have $c = \mathsf{com}(\mu'; r - (\mu' - \mu) r_u, s - (\mu' - \mu) s_v)$ for some $r_u, s_v \in \mathbb{Z}_p^{2m}$ (which may be specified in key generation). Thus we can create a perfectly hiding commitment and open it to any value we wish if we have the trapdoor key (r_u, s_v).

D.2 Plaintext Equivalence Proof [1]

We describe an adaption of the plaintext equivalence proof by Abe and Fehr [1] to demonstrate a NIZK proof to commitments to an identical message under different public keys.

Proof $P_{\mathsf{eq}}(c_1, c_2)$

Common Reference String : $\sigma = (\mathbf{f}, \mathbf{g}, \mathbf{h}, \mathsf{pk}_1, \mathsf{pk}_2)$ where $\mathsf{pk}_1 = (\mathbf{u}_1, \mathbf{v}_1, \mathbf{w}_1), \mathsf{pk}_2 = (\mathbf{u}_2, \mathbf{v}_2, \mathbf{w}_2) \leftarrow K_{\mathsf{bind}}(1^\lambda, \mathbf{f}, \mathbf{g}, \mathbf{h})$.
Statement : c, c' are commitments to μ under pk_1 and pk_2.
Prover's Input : (μ, r, s, r', s') so that $c_1 = (\mathbf{f}^r \mathbf{u}_1^\mu, \mathbf{h}^s \mathbf{v}_1^\mu, \mathbf{g}^{r+s} \mathbf{w}_1^\mu)$, $c_2 = (\mathbf{f}^{r'} \mathbf{u}_2^\mu, \mathbf{h}^{s'} \mathbf{v}_2^\mu, \mathbf{g}^{r'+s'} \mathbf{w}_2^\mu)$.
Proof :

$$\begin{aligned}
\pi_1 &= \mathbf{g}^{r-r'+r''} \\
\pi_2 &= \mathbf{f}^{r''} (\mathbf{u}_1^{-1} \mathbf{u}_2)^\mu \\
\pi_3 &= \mathbf{h}^{r''} (\mathbf{v}_1^{-1} \mathbf{v}_2)^\mu \\
\pi_4 &= \mathbf{g}^{r''} (\mathbf{w}_1^{-1} \mathbf{w}_2)^\mu
\end{aligned}$$

Send $\pi = (\pi_1, \pi_2, \pi_3, \pi_4)$ to the verifier.
Verifier : Check that

$$\begin{aligned}
& e(\mathbf{f}, \pi_1) = e(\mathbf{g}, c_{11} c_{21}^{-1} \pi_2) \\
& e(\mathbf{h}, c_{13} c_{23}^{-1} \pi_1^{-1} \pi_4) = e(\mathbf{g}, c_{12} c_{22}^{-1} \pi_3)
\end{aligned}$$

D.3 Range Proof

We describe an adaption of the well-known range proof by bit decomposition of the input adapted to the decision linear commitments setting.

Proof $P_{\mathsf{range}}(c, 2^\lambda)$

Common Reference String : $\sigma = (\mathbf{f}, \mathbf{g}, \mathbf{h}, \mathsf{pk})$ where $\mathsf{pk} = (\mathbf{u}, \mathbf{v}, \mathbf{w}) \leftarrow K_{\mathsf{bind}}(1^\lambda, \mathbf{f}, \mathbf{g}, \mathbf{h})$.
Let $[\mu]_j$ be the j^{th} bit of integer μ.
Statement : c is a commitment to μ under $(\mathbf{u}, \mathbf{v}, \mathbf{w})$ and $\mu \in [0, 2^\lambda - 1]$.
Prover's Input : μ, r, s so that $c = (\mathbf{f}^r \mathbf{u}^\mu, \mathbf{h}^s \mathbf{v}^\mu, \mathbf{g}^{r+s} \mathbf{w}^\mu)$.
Proof : For $0 \le j < \lambda$ let

$$\begin{aligned}
c_j &= (\mathbf{f}^{r_j} \mathbf{u}^{[\mu]_j}, \mathbf{h}^{s_j} \mathbf{v}^{[\mu]_j}, \mathbf{g}^{r_j+s_j} \mathbf{w}^{[\mu]_j}) \\
\pi_{1j} &= (\mathbf{u}^{2[\mu]_j-1} \mathbf{f}^{r_j})^{r_j} \\
\pi_{2j} &= (\mathbf{v}^{2[\mu]_j-1} \mathbf{h}^{s_j})^{s_j} \\
\pi_{3j} &= (\mathbf{w}^{2[\mu]_j-1} \mathbf{g}^{(r_j+s_j)})^{(r_j+s_j)}
\end{aligned}$$

Let $\pi' = \mathbf{g}^{r - \sum_{j=0}^{\lambda-1} 2^j \cdot r_j}$.
Send $(c_j, \pi_{1j}, \pi_{2j}, \pi_{3j})_{0 \le j < \lambda}$, and π'.

Verifier : For $0 \leq j < \lambda$ check that

$$\begin{aligned}
e(\mathbf{f}, \pi_{1j}) &= e(c_{1j}, c_{1j}\mathbf{u}^{-1}) \\
e(\mathbf{h}, \pi_{2j}) &= e(c_{2j}, c_{2j}\mathbf{v}^{-1}) \\
e(\mathbf{g}, \pi_{3j}) &= e(c_{3j}, c_{3j}\mathbf{w}^{-1}) \\
e(\mathbf{f}, \pi') &= e(\mathbf{g}, c_1 \cdot \prod_{j=0}^{\lambda-1}(c_{1j})^{-2^j}) \\
e(\mathbf{h}, c_3 \cdot \prod_{j=0}^{\lambda-1}(c_{3j})^{-2^j} \cdot \pi'^{-1}) &= e(\mathbf{g}, c_2 \cdot \prod_{j=0}^{\lambda-1}(c_{2j})^{-2^j})
\end{aligned}$$

Theorem 8. *The above range proof is perfectly complete, perfectly sound and is computational zero knowledge if the Decision Linear assumption holds. The proof consists of* $2 + 12m\lceil \log_2 \mu \rceil$ *group elements.*

Completeness It is straightforward to check that the verification equations hold if the prover is honest.

Zero Knowledge Under the decision linear assumption, the common reference string σ may be simulated so that $(\mathbf{u}, \mathbf{v}, \mathbf{w})$ form a linear tuple, i.e., where $\mathbf{u} \in G^2, \mathbf{v} \in G^2, \mathbf{w} = \mathbf{u}^{x^{-1}}\mathbf{v}^{y^{-1}}$. The simulator sets $\mathbf{f} = \mathbf{g}^x, \mathbf{h} = \mathbf{g}^y$: $x, y \leftarrow \mathbb{Z}_p^{2m}$ and outputs (σ, τ) where $\sigma = (\mathbf{f}, \mathbf{h}, \mathbf{g}, \mathbf{u}, \mathbf{v}, \mathbf{w})$ and $\tau = (x, y)$. The simulator chooses $r_j, s_j \in \mathbb{Z}_p^{2m}$ and computes $c_j = (c_{1j}, c_{2j}, c_{3j}) = (\mathbf{f}^{r_j}, \mathbf{h}^{s_j}, \mathbf{g}^{r_j+s_j})$. It sets $\pi_{1j} = (c_{1j}^2 \cdot \mathbf{f}^{-r_j} \cdot \mathbf{u}^{-1})^{r_j}, \pi_{2j} = (c_{2j}^2 \cdot \mathbf{h}^{-s_j} \cdot \mathbf{v}^{-1})^{s_j}$ and $\pi_{3j} = (c_{3j}^2 \cdot \mathbf{g}^{-(r_j+s_j)} \cdot \mathbf{w}^{-1})^{(r_j+s_j)}$. It sets $\pi' = (c_1 \cdot (\prod_{j=0}^{\lambda-1}(c_{1j})^{2^j})^{-1})^{x^{-1}}$. By inspection, $\pi_{1j}, \pi_{2j}, \pi_{3j}$ and π' are distributed identically as in the real protocol with respect to σ and $(c_j)_j$ so computational zero knowledge follows.

Soundness There exists r_j and s_j so that $c_j = \mathsf{com}(\mu_j; r_j, s_j)$ for some $\mu_j \in \mathbb{Z}_p$. Moreover, by the perfect binding property of the commitment scheme, these are all unique. Then valid π_{1j} implies that $e(\mathbf{f}, \pi_{1j}) = e(\mathbf{u}, \mathbf{u})^{\mu_j(\mu_j-1)}e(\mathbf{f}, \mathbf{u}^{r_j(2\mu_j-1)})$ $e(\mathbf{f}, \mathbf{f})^{r_j^2}$. If $\mathbf{u} \notin \text{span}(\mathbf{f})$, it follows that $\mu_j(\mu_j - 1) = 0$, thus $\mu_j = 0$ or $\mu_j = 1$. Similarly valid π_{2j} and $\mathbf{v} \notin \text{span}(\mathbf{h})$, and valid π_{3j} and $\mathbf{w} \notin \text{span}(\mathbf{g})$ imply the same result. On the other hand the perfect binding instantiation of the commitment scheme implies that $(\mathbf{u}, \mathbf{v}, \mathbf{w})$ is a non-linear tuple, so one of $\mathbf{u} \notin \text{span}(\mathbf{f})$,$\mathbf{v} \notin \text{span}(\mathbf{h})$, or $\mathbf{w} \notin \text{span}(\mathbf{g})$ holds. It follows that $\mu_j = 0 \vee \mu_j = 1$ for $0 \leq j < \lambda$. The existence of valid π' implies that r' and s' exist satisfying the equations $\pi' = \mathbf{g}^{r'}, c_1 = \mathbf{f}^{r'+\sum_{j=0}^{\lambda-1} 2^j r_j}, c_2 = \mathbf{h}^{s'+\sum_{j=0}^{\lambda-1} 2^j s_j}, c_3 \cdot \prod_{j=0}^{\lambda-1}(c_{3j})^{2^j} \cdot \pi'^{-1} = g^{s'}$. This implies $c_3 = \mathbf{w}^{\mu}g^{r'+s'}$: $\mu = \sum_{j=0}^{\lambda-1} \mu_j 2^j$. Hence $\mu \in [0, 2^\lambda - 1]$.

E Distributed Key Generation Sub-Protocols

E.1 Conditional Gate

Schoenmakers et al. [38] describe a protocol by which a certain multiplication gate may be distributed across N parties with shares of an additively homomorphic threshold cryptosystem. The first input to the gate is from a dichotomous (two-valued) domain while the second input is unrestricted. Let $\mathcal{F}_{\mathsf{COND}}$ be the ideal functionality with the following behaviour.

– On input $[x]$ and $[y]$ returns an encryption of $[xy]$ if $x \in \{-1, = 1\}$ and $\perp$ otherwise.

Notation: In this section we use $\oplus$, $\otimes$ and $\ominus$ to mean homomorphic addition, multiplication and subtraction respectively.

For properly formed ciphertexts $[x]$ and $[y]$ let $[x] \star [y]$ denote the output of $\mathcal{F}_{\mathsf{COND}}([x], [y])$. It is shown in [38] that players may compute an encryption of xor-sum of bit-valued inputs x and y according to the following sequence of transformations $[x'] \leftarrow 2 \otimes [x] \ominus [1] : x' = 2x - 1, [x'y] \leftarrow [x'] \star [y], [x \oplus y] \leftarrow [x] \ominus [x'y]$. The protocol is given in Figure 8, with decryption in Figure 11. This sequence forms the basis for our protocol in Section E.2 which computes the private xor-sum of a number of encrypted inputs.

Theorem 1 [38] states that for all input pairs $([x], [y]) : x \in \{-1, 1\}$ and any PPT adversary A corrupting at most T players in an execution of the above protocol, there exists a simulator S that interacts with the ideal world functionality computing the conditional gate, $\mathcal{F}_{\mathsf{COND}}$, and outputs a transcript which is computationally indistinguishable from that obtained by execution in the real world. This simulator is described in Figure 9.

Protocol π_{COND} ([38])
Common Input : Let $[x], [y]$ denote encryptions with $x \in \{-1, 1\} \subseteq \mathcal{M}$ and $y \in \mathcal{M}$.
Private Input : Player P_i holds a share of the secret key, sk_i.

Let $x_0 = x$ and $y_0 = y$.

1. Player P_i takes $[x_{i-1}]$ and $[y_{i-1}]$ as input and broadcasts a commitment $\langle\langle s_i \rangle\rangle$ with $s_i \in_R \{-1, 1\}$. Then P_i computes $[x_i] = [x_{i-1}] \otimes s_i \oplus [0]$ and $[y_i] = [y_{i-1}] \otimes s_i \oplus [0]$ together with $\varepsilon_i \leftarrow P_{\mathsf{mul}}(\langle\langle s_i \rangle\rangle, [x_{i-1}], [x_i]), P_{\mathsf{mul}}(\langle\langle s_i \rangle\rangle, [y_{i-1}], [y_i])$. If P_i fails to complete this step it is discarded immediately.
2. The parties jointly decrypt $[x_n]$ to obtain x_n. If decryption fails because the number of correct shares is insufficient, the entire protocol is aborted. If decryption fails because $x_n \notin \{-1, 1\}$ each party P_i is required to broadcast a proof that $s_i \in \{-1, 1\}$. Parties failing to do so are are discarded, and the protocol is restarted. Given x_n and $[y_n]$ and encryption $[x_n y_n]$ is computed publicly.

Fig. 7. Protocol π_{COND} computing the conditional gate.

E.2 Private Sum Modulo Two

Protocol π_{SUM}
Common Input : $[u_1], \ldots, [u_n]$ under Enc_1 and $u_i \in \{0,1\}^\lambda$ for $1 \leq i \leq n$.
Private Input : P_i holds u_i and randomness r_i of $[u_i]$ and share of secret key, sk_i

Player P_i

1. Let $u_i = \sum_{j=0}^{\lambda-1} u_{ij} 2^j$ where $u_{ij} \in \{0,1\}$. Broadcast $c_{ij} = [u_{ij}]$ under randomness r'_{ij} and $\varepsilon_{ij} \leftarrow P_{\mathsf{bit}}(c_{ij})$.
2. Let $C_0 = \ldots = C_{\lambda-1} = \mathsf{Enc}_1^*(0)$.
3. If any of the ε_{ij} do not pass verification output $\perp$. Otherwise, for $1 \leq k \leq N$, for $0 \leq j < \lambda$, let:
   (a) $c'_{kj} = 2 \otimes c_{kj} \ominus \mathsf{Enc}_1^*(1)$
   (b) $c''_{kj} = \pi_{\mathsf{COND}}(\mathsf{sk}_i, c'_{kj}, C_j)$
   (c) $C_j = c_{kj} \ominus c''_{kj}$
4. Output $\boldsymbol{C} = (C_0, \ldots, C_{\lambda-1})$.

Fig. 8. Protocol π_{SUM}.

Theorem 9. *Protocol π_{SUM} securely computes encryption switching in the $\mathcal{F}_{\mathsf{COND}}$-hybrid model against statically chosen adversaries if Π_{bit} is a secure NIZK proof system.*

Proof. Let $(c^*_{ij}, \varepsilon^*_{ij}, C^*_j, {c'_{ij}}^*, {c''_{ij}}^*)_{1 \leq i \leq N}$ be the output of the adversary interacting with the honest players in the real protocol conditional on the event $C^*_j = \tilde{C}_j$. Then by the assumption that at most T players are corrupted, ${c''_{ij}}^*$ is a an encryption of the conditional product of the plaintexts contained in ${c'_{ij}}^*$ and C^*_j for $j \neq N$. Moreover, by the soundness of the zero knowledge proof for $\mathcal{R}_{\mathsf{bit}}$, ciphertext c^*_{ij} is a an encryption of u^*_{ij} where $u^*_{ij} \in \{0,1\}$. Let ${u'_{ij}}^* = 2 * u^*_{ij} - 1$. Now the relations $C_j = c_{kj} \ominus c''_{kj}$ and $C_j = \sum_{i=1}^{k} u^*_{ij} \prod_{l=1}^{k} (-1)^k {u'_{lj}}^*$ hold at the k^{th} invocation of Step 3c. This holds by inspection for $k < N$. For $k = N$ this follows since $c_{Nj} = \tilde{C}_j \ominus \tilde{C}''_j$ implies $c_{Nj} = C_j \ominus C''_j$ where $C''_j = C'_j \star \tilde{C}_j$: $C'_j = 2 \otimes C_j - \mathsf{Enc}_1^*(1)$, exploiting the fact that C_j and $\tilde{C}_j$ are known to be bit encryptions. Next $c_{Nj} = C_j \ominus C''_j$ implies $C_j = c_{Nj} \ominus C'_j \star \tilde{C}_j$ and hence that $C_j = c_{Nj} \ominus c''_{Nj}$. Also $C_j = [u^*_{Nj}] - [(2 * u^*_{Nj} - 1) \sum_{i=1}^{N-1} u^*_{ij} \prod_{l=i+1}^{N-1} (-1)^{N-1} {u'_{lj}}^*] = [u^*_{Nj} - {u'_{Nj}}^* \sum_{i=1}^{N-1} u^*_{ij} \prod_{l=i+1}^{N-1} (-1)^{N-1} {u'_{lj}}^*] = [\sum_{i=1}^{N} u^*_{ij} \prod_{l=i+1}^{N} (-1)^N {u'_{lj}}^*]$. A hybrid argument thus implies that if the adversary could distinguish the simulated view, using the simulated expression for c_{Nj} in place of the real, they could break the semantic security of Enc_1 which by assumption is impossible.

```
Simulator for $\pi_{\mathsf{SUM}}$
Input : $([u_1], \ldots, [u_N], \boldsymbol{C} = (\tilde{C}_0, \ldots, \tilde{C}_{\lambda-1})$

– Let $B$ be the set of corrupted players.

1. Let $(\hat{\sigma}_n, \tau_N) \leftarrow S_1(1^\lambda)$
2. Perform Steps 1–3 on behalf of honest players $P_i : i \in \overline{B} \backslash \{N\}$ except that $c''_{kj}$
   is computed as $\mathcal{F}_{\mathsf{COND}}(c'_{kj}, C_j)$.
3. For $0 \leq j < \lambda$
   (a) Let $\tilde{C}'_j = 2 \otimes \tilde{C}_j \ominus \mathsf{Enc}^*_1(1), \tilde{C}''_j = \mathcal{F}_{\mathsf{COND}}(\tilde{C}'_j, C_j)$.
   (b) Publish $c_{Nj} = \tilde{C}_j \ominus \tilde{C}''_j$ and $\varepsilon_{Nj} \leftarrow S_2(\tau_N, c_{Nj})$.
   (c) Let $c'_{Nj} = 2 \otimes c_{Nj} \ominus \mathsf{Enc}^*_1(1)$.
   (d) Let $c''_{Nj}$ be the output of $\mathcal{F}_{\mathsf{COND}}(c'_{Nj}, C_j)$.
```

Fig. 9. Simulator for protocol π_{SUM}.

E.3 Decryption Protocol

Lemma 2. *Protocol π_{DEC} unconditionally computes encryption switching against statically chosen adversaries.*

Proof. As we have T honest players, $|J| \geq T$. Validation of the pairings of submitted shares of $P_i \in J$ implies that $\mathrm{dlog}_{z_2} z_{2i} = \mathrm{dlog}_{g_{\mathsf{pke}}} \prod_{j \in Q} A_{ij} = x_i$, $\mathrm{dlog}_{z_3} z_{3i} = \mathrm{dlog}_{h_{\mathsf{pke}}} \prod_{j \in Q} A'_{ij} = x'_i$ and $\mathrm{dlog}_{h_{\mathsf{pke}}} \prod_{j \in Q} B_{ij} = b_i$ hold. Therefore there exists $R \subseteq J, |R| \geq T$ for which reconstruction coefficients $\Lambda_{j,R}$ may be applied to $(z_{j2}, z_{j3}, z_{j4})_j$ to compute $\pi_t(z_1, z_2, z_3, z_4)$ and hence M. As $J \subset Q$ defined by Protocol 1, there is at most a negligible probability P_i does not submit correct (z_{i2}, z_{i3}, z_{i4}) so that $M \neq m$.

F Parameter Selection for Distributed Key Generation

The following theorem characterises the range of parameters under which key generation may proceed securely.

Lemma 3. *Suppose that $2 \log_2 N + \log_2 \ell + 2\lambda_A + \lambda < \lambda_B$. Then for each k the distribution of γ_k produced in Step 7 is statistically independent of the contributions by the honest players to the k^{th} chunk of xx', i.e., $(xx' - [xx' \bmod c_B^{k+1}])/c_B^k$, except with error at most $2^{-\lambda}$.*

Proof. Let $SD(\cdot, \cdot)$ denote the statistical (i.e., total-variation) distance between random variables. Let $C \subset Q$ be the set of players corrupted by the adversary. For fixed k, let $a_{ij} = \sum_{f+g=k} \alpha_{if} \alpha'_{jg}$ and $b_i = \beta_{ik}$. Thus $\gamma_k = \sum_{i,j \in Q} \sum_{f+g=k} \alpha_{if} \alpha'_{jg} + \sum_{i \in Q} \beta_{ik} = \sum_{i,j \in Q} a_{ij} + \sum_{i \in Q} b_i = \sum_{i,j \in Q \backslash C} a_{ij} + \sum_{(i,j) \in C \times Q \cup Q \times C} a_{ij} + \sum_{i \in Q \backslash C} b_i + \sum_{i \in C} b_i = X + \overline{X} + Y + \overline{Y}$. Clearly $|X| \leq (|Q| - T)^2 \cdot \ell \cdot 2^{2\lambda_A}$. On the

Protocol π_{DEC}
Common Input : $C = (z_1, z_2, z_3, z_4)$ output of $\mathsf{Enc}_{\mathsf{tgt}}$.
Private Input : P_i holds a share of secret key, sk_i

Player P_i :

1. Submit $(i, (z_{i1}, z_{i2}, z_{i3}, z_{i4}))$.
2. Let $J = \{P_j\}_j$ for which

$$e(\prod_{j \in Q} A_{ij}, h_{\mathsf{pke}}) = z_{i2}$$
$$e(g_{\mathsf{pke}}, \prod_{j \in Q} A'_{ij}) = z_{i3}$$
$$e(g_{\mathsf{pke}}, \prod_{j \in Q} B_{ij}) = z_{i4}$$

   simultaneously hold per Equation 2, Protocol 1.
3. Outputs $M \leftarrow \log_{\pi_t(e(\mathbf{g},\mathbf{h}))}(\pi_t(z_1, z_2, z_3, z_4))$ as in Theorem 2.

Fig. 10. Protocol π_{DEC}.

Simulator for π_{DEC}
Input : C, m

– Let $B : |B| \geq T$ be honest.

1. Perform the above steps for honest parties in B.
2. If P_j does not satisfy the above checks B extracts x_j, x'_j, b_j and computes (z_{j2}, z_{j3}, z_{j4}).
3. Output $\perp$ if $m \neq \log_{\pi_t(e(\mathbf{g},\mathbf{h}))}(\pi_t(z_1, z_2, z_3, z_4))$.

Fig. 11. Simulator for protocol π_{DEC}.

other hand, $Y \overset{d}{\approx} \mathcal{N}((|Q|-T) \cdot 2^{\lambda_b - 1}, (|Q|-T) \cdot \frac{2^{2\lambda_B}}{12})$. Then $SD(X, X+Y) \geq 1 - 2^{-\lambda}$ holds when

$$\begin{aligned}((|Q|-T)^2 \cdot \ell \cdot 2^{2\lambda_A} \cdot 2^{\lambda})^2 &\leq \frac{(|Q|-T) \cdot 2^{2\lambda_B}}{12} \\ \Leftrightarrow (|Q|-T)^3 \cdot \ell^2 \cdot 2^{4\lambda_A} \cdot 2^{2\lambda} &\leq \frac{2^{2\lambda_B}}{12} \\ \Leftarrow N^3 \cdot \ell^2 \cdot 2^{4\lambda_A} \cdot 2^{2\lambda} &\leq 2^{2\lambda_B}. \end{aligned} \tag{3}$$

By non-malleability of Π_{range} the random variables $\overline{X}$ and $\overline{Y}$ are independent of X and Y, so $SD(X, X+\overline{X}+Y+\overline{Y}) \geq SD(X, X+Y) \geq 1-2^{-\lambda}$. Taking logarithms of both sides of Equation 3 and multiplying by one-half yields the result.

G Proving Correct Decryption

Protocol π_{SWITCH} uses a proof of correct decryption, which follows from the solution proposed in [39] for instance.

Protocol π_{CD} ([39])
Input : The tuple $\{(d, d_i, v, v_i; s_i) \mid d_i \leftarrow d^{s_i}\}$.
Announcement : $\Sigma.\mathsf{ann}(d, d_i, v, v_i; s_i) := u \in_R [0, 2^{\log_2 N + \lambda}]; a = d^u; b = v_i^u;$ **return** $(a, b; u)$
Response : $\Sigma.\mathsf{res}(d, d_i, v, v_i; a, b; u, c) := r := u + cs_i;$ **return** r
Verification : $\Sigma.\mathsf{ver}(d, d_i, v, v_i; a, b; c; r) := d^r \overset{?}{=} a(d_i)^c \wedge v^r \overset{?}{=} b(v_i)^c$
Extractor : $\Sigma.\mathsf{ext}(d, d_i, v, v_i; a, b; c; c'; r; r') :=$ **return** $(r - r')/(c - c')$
Simulator : $\Sigma.\mathsf{sim}(d, d_i, v, v_i; c) := r \in_R [0, 2^{\log_2 N + \lambda}];$ **return** $(d^r(d_i)^{-c}, v^r(v_i)^{-c}; c; r)$

H Definition of Universally Verifiable computation

H.1 Universally Verifiable Encryption Switching

Our ideal model for universally verifiable encryption switching is derived from [39], Process 5 and detailed in Figure 12.

H.2 Universally Verifiable Function Evaluation

We will show how use distributed encryption switching to achieve universally verifiable secure function evaluation of any function representable as a polynomial size arithmetic circuit over a prime field $\mathbb{Z}_p$. We assume that the inputs

Ideal party for verifiable encryption switching

```
Compute encryption switching for $\mathcal{R}$ with corrupted parties $B$; $\mathcal{V}$ learns encryption in target group.
Common Input: $(\mathsf{pk}_1, c, \bar{c})$, threshold $t$.
for $i \in N \backslash B$ do
    $sk_i := recv(P_i)$
end for                                                        $\triangleright$ Honest inputs.
$\{sk_i\}_{i \in B} :=$ recv($\mathcal{S}$)                    $\triangleright$ Corrupted inputs.
if $|B| \geq t$ then
    Send $r_i$ to $\mathcal{S}$ for all $i$.   $\triangleright$ Threshold corrupt.   $\triangleright$ Computation phase.
    $E = \mathsf{Enc}_{tgt}(m)$
end if
if $\mathcal{R} \notin B$ then   $\triangleright$ honest $\mathcal{R}$; adversary learns encryption, may block result.
    send($E, \mathcal{S}$)
    if $|N \backslash B| < t$ and recv$\mathcal{S} = \perp$ then
        send($\perp, \mathcal{R}$).
    end if
else   $\triangleright$ Corrupted $\mathcal{R}$. Adversary learns output, may block result to $\mathcal{V}$.
    send($m, \mathcal{S}$); $s =$ recv$\mathcal{S}$
    if $s = \perp$ then
        $R := \perp$
    else$R = \mathsf{Enc}_{tgt}(m)$.
    end if                                                     $\triangleright$ Proof phase.
    if $\mathcal{V} \notin C$ then send($R, \mathcal{V}$).
end if
```

Fig. 12. Ideal party for verifiable π_{SWITCH}.

to the function are T-threshold encrypted under the one time homomorphic encryption scheme of Section 2. Let f be an arbitrary function with domain $\mathbb{Z}_p^\omega$ and output in $\mathbb{Z}_p$ with a representation as a polynomial size arithmetic circuit C_f over $\mathbb{Z}_p$. Let the input wires of C_f be indexed $1, \ldots, \omega$, the internal wires be indexed $\omega + 1, \ldots, |C_f| - 1$ and the output wire have index $|C_f|$. Every gate has fan-in two and is indexed (u, v, w) where u, v are indices of the input wires and w is the index of the output wire.

H.3 The Protocol

Input Ciphertexts $c_1, \ldots, c_\omega$ by Enc_1. Circuit C_f representing function f.
Private input Share of secret key sk_i.
Evaluation

Addition Gates

- Player P_i computes locally $c_w = c_u \oplus_1 c_v$, where $c_u = \mathsf{Enc}_1(m_u)$ and $c_v = \mathsf{Enc}_1(m_v)$ are the ciphertexts associated to input wires u and v.

Multiplication Gates

- Player P_i computes locally $c'_w = c_u \boxtimes_1 c_v$.
- Player P_i participates in the switching protocol to produce $c_w \leftarrow \mathsf{Switch}(\mathsf{pk}, c'_w, \mathsf{sk}_i)$.

Output Gate

- Player P_i participates in distributed decryption to produce $m_{|C_f|} \leftarrow \mathsf{Decrypt}_2(\mathsf{sk}_i, c'_{|C_f|})$.

Fig. 13. Protocol π_{EVAL} for universally verifiable function evaluation.

Theorem 10. *Protocol π_{EVAL} achieves universally verifiable function evaluation against any static adversary corrupting a minority of parties assuming a secure distributed encryption scheme $\Pi_1 \rightleftharpoons \Pi_2$ equipped with the T-threshold decryption structure. The round complexity is proportional to the depth of C_f and communication complexity is $O(|C_f| \cdot T)$.*

Proof. We prove the result by a sequence of intermediate games. Specifically we will prove that the advantage of any adversary $\mathsf{Adv}^{\mathsf{EVAL}}(\lambda)$ in distinguishing the view of corrupted parties in the real protocol from the simulation (involving inputs chosen uniformly at random and without secret keys) is at most $(\mathsf{Adv}^{\mathsf{SWITCH}}_{\mathbb{A}_{T\text{-Th}}}(\lambda) + \mathsf{Adv}^{\mathsf{Enc}_1}(\lambda)) \cdot |C_f| + \mathsf{Adv}^{\mathsf{DEC}}(\lambda)$ where $\mathsf{Adv}^{\mathsf{SWITCH}}_{\mathbb{A}_{T\text{-Th}}}(\lambda)$, $\mathsf{Adv}^{\mathsf{DEC}}(\lambda)$ denote the simulation error in Protocols π_{SWITCH} and π_{DEC} and $\mathsf{Adv}^{\mathsf{Enc}_1}(\lambda)$ denotes the distinguishing advantage against the one-time homomorphic scheme of Section 2.

Game G_0: This is the actual execution where the adversary interacts with the real switching challenger, running the setup algorithm $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Setup}(1^\lambda)$ and generates all shares $(\mathsf{sk}_i)_i$ according to the T-threshold access structure.

Game G_1: In this game the challenger chooses all inputs $r_1, \ldots, r_\omega$ uniformly at random in $\mathcal{M}_1$ and generates corresponding ciphertexts $c_1, \ldots, c_\omega$. It keeps track of the values r_w produced at each gate (u, v, w) during the subsequent computation.

Game G_2: In this game the challenger simulates the output of addition gates by computing $c_w \leftarrow c_u \oplus_1 c_v$ for itself on behalf of all honest parties.

Game G_3: In this game the challenger simulates all secret key shares by running the simulator $\mathcal{S}_1$ of the ideal switching experiment. Let (u, v, w) be a multiplication gate. Every call to π_{SWITCH} is answered by having the adversary interact with $\mathcal{S}_2$ on $(c'_w, \hat{c}_w)$.

Case 1 : $w \neq |C_f|$. Let $c'_w \leftarrow c_u \otimes c_v$ and let $\hat{c}_w$ be a uniformly random encryption of $r_u \cdot r_v$ by Enc_1.

Case 2: $w = |C_f|$. In this case the simulator chooses a uniformly random encryption of $m_{|C_f|}$ for $\hat{c}_w$. Let P_N be honest. The challenger runs the simulator for π_{SWITCH} but computes $d_N = \hat{c}_1(\prod_{i \neq T} d_i)^{-1}$ and $\xi_N \leftarrow \Sigma.\mathsf{sim}(d, d_N, \mathsf{pk}, \mathsf{vk}_N)$.

Game G_4: The challenger peforms a simulated decryption on $\hat{c}_{|C_f|}$ and passes the output to the adversary.

A hybrid argument implies that games G_0 and G_1 can be distinguished with probability at most $|C_f| \cdot \mathsf{Adv}^{\mathsf{Enc}_1}(\lambda)$. On the other hand G_1 and G_2 are identical from the point of view of the adversary. It is easily seen inductively that output of $\mathcal{S}_2$ on the twin ciphertext pair $(c'_w, \hat{c}_w)$ must be indistinguishable from the real-world output using $\mathsf{Enc}_1/\mathsf{Enc}_2$ up to the probability of an early abort, bounded by $|C_f| \cdot \mathsf{Adv}^{\mathsf{SWITCH}}_{\mathbb{A}_{T\text{-Th}}}(\lambda)$. The case $w = |C_f|$ follows from the correct distribution of $c_{|C_f|}$. Lastly the simulation error in distributed decryption is $\mathsf{Adv}^{\mathsf{DEC}}(\lambda)$.

As an immediate consequence of Theorem 10 we have

Corollary 11 *Let $\mathcal{C}_\lambda \in \mathsf{P/Poly}$ be a family of size-bounded circuits and suppose there exists an efficient attacker that distinguishes the N-party distributed evaluation of $\mathcal{C}_\lambda$ (Figure 6) with non-negligible advantage ϵ'. Then there exists an attacker who breaks the External l-Symmetric Cascade Assumption with probability at least*

$$\frac{\epsilon' - \lambda^c \cdot \mathsf{Adv}^{\mathsf{SWITCH}}_{\mathbb{A}_{T\text{-Th}}}(\lambda)}{\lambda^c}$$

with at most $O(\lambda^c)$ combined group/pairing operations over $\mathbb{G}_1$ where c is an absolute constant.

I Cost Comparison to [20]

Table 2 compares the total arithmetic operations used in our scheme to [20]. Given the pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_t$ let op_{ExpDH} be a Diffie-Hellman exponentiation in $\mathbb{G}_1$. Let op_{ExpCF} be a Catalano-Fiore exponentiation. Let op_{AAdd}

and op_{AMult} be addition and multiplication operations respectively in the algebra $\mathbb{A}_q = \mathbb{Z}_q[X]/\Phi_m(X)$. Assume secret sharing threshold $T = N/2$ and chunk parameter $\ell = 8$.

Ours	$(8c(k^2 - k)M + 6kcN) \times op_{ExpDH} + (128N^2 + 32N) \times op_{ExpCF}$
[20]	$7c(k^2 - k)M(N \times op_{AAdd} + 2 \times op_{AMult})$

Table 2. Cost comparison to [20].

J Protocol and implementation details for IRV counting

This section describes the detailed protocol for IRV counting.

Theorem 12. *Let ϵ_{IS} be the maximal distinguishing advantage of adversary $\mathcal{A}$ in the ideal switching experiment. Then Protocol 4 securely realises universally verifiable IRV tallying against statically chosen adversaries except with error probability at most $\binom{k}{2} \cdot \epsilon_{\mathsf{IS}}$.*

Proof. Our proof proceeds by a hybrid argument following Claim J. Define $H_I^{(l)}$ to be the experiment in which on the l^{th} round of tallying during the first I invocations of the for loop on line 8 the adversary instead interacts with $\mathcal{B}_2$ on inputs $\mathsf{pk}, (\mathsf{sk}_i)_{i \in \overline{B}}, \boldsymbol{\pi}_j^{(l)}, \boldsymbol{\pi}'^{(l)}_j$ and with the ideal decryption functionality, $\mathcal{F}_{\mathsf{DEC}}$, on line 17. Note that $H_l^{(l)}$ is identical to $H_1^{(l+1)}$, in which all encryption switching during the first l rounds of tallying is generated from the simulator who is given access to key shares of honest parties and public intermediate products. Thus, given Claim J, we may conclude the proof by noting that the maximal distinguishing advantage between real and simulated protocols is $\mathrm{Adv}_{\mathcal{A}}(\lambda) = |\Pr[\mathcal{A}(H_1^{(1)}) = 1] - \Pr[\mathcal{A}(H_k^{(k)}) = 1]|$. In particular, it follows that $\epsilon \le \sum_{l=1}^{k} |\Pr[\mathcal{A}(H_1^{(l-1)}) = 1] - \Pr[\mathcal{A}(H_1^{(l)}) = 1]| \le \sum_{l=1}^{k} \sum_{I=1}^{l} |\Pr[\mathcal{A}(H_{I-1}^{(l)}) = 1] - \Pr[\mathcal{A}(H_I^{(l)}) = 1]| \le \binom{k}{2} \cdot \epsilon_{\mathsf{IS}}$.

Claim. In the $\mathcal{F}_{\mathsf{DEC}}$-hybrid model for any PPT adversary $\mathcal{A}$ who distinguishes between $H_{I-1}^{(l)}$ and $H_I^{(l)}$ with probability ϵ, there exists an adversary who distinguishes between real and ideal switching with probability $\epsilon_{\mathsf{IS}} := \epsilon$.

Proof. This is immediate from the perfect emulation of honest by parties by $\mathcal{O}_{\overline{B}}$.

Implementation Details A proof-of-concept implementation of the IRV counting protocol was made in Python 2.7 using the PPAT library[6] for group operations, based on a BN curve [4] with a prime modulus of 256 bits. The implementation did not include the construction or checking of proofs, and ran as a single

[6] `https://github.com/ecuvelier/PPAT`

party. It would be straightforward to include multiple parties with the addition of the appropriate communication code.

The election for Albury was settled after just one round, and took just under two hours. The election for Auburn took 4 rounds and completed in a little over 15 hours. In both cases the intermediate and final results were compared with the official results to ensure accuracy of our counting algorithm.

K Proofs of Auxiliary Results

Proof of Theorem 2

Correctness It suffices to compute the projection π_t of any element in $G_t = \mathbb{G}_t^4$ which will follow from the following elementary results concerning the correctness of γ, x, x' and b shown in Propositions 3 and 4 below. Any qualified set $R \subseteq Q$ may compute $\pi_t(z_1, z_2, z_3, z_4)$ as follows

$$\begin{aligned}\pi_t(z_1,z_2,z_3,z_4) &= z_1 z_2^x z_3^{x'} z_4^{xx'} = \frac{z_1 z_2^x z_3^{x'} z_4^{xx'+b}}{z_4^b} \\ &= \frac{z_1 z_2^x z_3^{x'} z_4^{\gamma}}{z_4^b} \\ &= \frac{z_1 z_2^{\sum_{j\in R} \Lambda_{j,R}(\sum_{i\in Q} s_{ij})} z_3^{\sum_{j\in R} \Lambda_{j,R}(\sum_{i\in Q} s'_{ij})} z_4^{\gamma}}{z_4^{\sum_{j\in R} \Lambda_{j,R}(\sum_{i\in Q} t_{ij})}} \\ &= \frac{z_1 z_2^{\sum_{j\in R} \Lambda_{j,R} x_j} z_3^{\sum_{j\in R} \Lambda_{j,R} x'_j} z_4^{\gamma}}{z_4^{\sum_{j\in R} \Lambda_{j,R} b_j}}\end{aligned}$$

In particular this expression may be computed from individual shares x_j, x'_j and b_j as the product $\frac{z_1 \prod_{j\in R}(z_2^{x_j})^{\Lambda_{j,R}} (z_3^{x'_j})^{\Lambda_{j,R}} z_4^{\gamma}}{\prod_{j\in R}(z_4^{b_j})^{\Lambda_{j,R}}}$.

Resilience It is trivial to verify the correctness of any submitted share using $(A_i, A'{}_i, B_i, B'{}_i)_{i\in Q}$, according to Equation 2. Thus we have resilience against an arbitrary number of misbehaving participants during the key reconstruction phase.

Privacy We show the existence of a polynomial time simulator that interacts with the ideal world CF key generation functionality and generates a distribution indistinguishable to that produced by an actual run of the protocol by honest players. In the following the proof of security proceeds in the ideal world CF key generation hybrid model.

We argue the simulation is good. First note that no adversary can distinguish a real common reference string σ_N from the simulated common reference string $\hat{\sigma}_N$ by the zero knowledge property of Π_{range} and Π_{eq}. We next note that steps 1-3 of the simulation are indistinguishable from the corresponding steps run in the real protocol. to see this, first note that all parties have access to the broadcast values V_i, V'_i and W_i, thus it is impossible for a honest player

not to be included in the qualified set decided in Step 3, i.e, $\overline{B} \subseteq Q$. On the other hand for player $P_j \in B \cap Q$, the simulator receives at least T shares, namely those corresponding to the honest players in Q, enabling recovery of s_j, s'_j and t_j. Thus the simulator can compute s_j, s'_j and t_j for all players $P_j \in Q$.
Let $A^*_i, B^*_i, A'^*_i, B'^*_i$ be the commitments broadcast in Step 4 of the real protocol and s^*_{ij},${s'_{ij}}^*$ and t^*_{ij} be the shares of secrets s^*_i, ${s'_i}^*$ and t^*_i sent by P_i to P_j in Step 1, where $P_i, P_j \in Q$. In that case we have that $A^*_i, B^*_i, A'^*_i, B'^*_i$ are chosen uniformly at random in $\mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{G}_2$ subject to the constraints that $y_{\mathsf{pke}} = \prod_{i \in Q} A^*_i[1], z_{\mathsf{pke}} = \prod_{i \in Q} B^*_i[1], y' = \prod_{i \in Q} B^*_i[1], z' = \prod_{i \in Q} {B'_i}^*[1]$, i.e. the $|Q|$-wise product of first components is fixed to y_{pke}, z_{pke}, y', z'. Similarly $(s^*_{ij})_{P_j \in Q \setminus \{N\}}$, $({s'_{ij}}^*)_{P_j \in Q \setminus \{N\}}$ and $(t^*_{ij})_{P_j \in Q \setminus \{N\}}$ are sampled uniformly at random. On the other hand, we have $\hat{A}_N[1] = y_{\mathsf{pke}} \cdot \prod_{i \in Q \setminus \{N\}} (\hat{A}_i[1])^{-1}$, $\hat{B}_N[1] = z_{\mathsf{pke}} \cdot \prod_{i \in Q \setminus \{N\}} (\hat{B}_i[1])^{-1}$, $\hat{A}'_N[1] = y' \cdot \prod_{i \in Q \setminus \{N\}}$ $(A'_i[1])^{-1}$, $\hat{B}'_N[1] = z' \cdot \prod_{i \in Q \setminus \{N\}} (B'_i[1])^{-1}$ and $\hat{A}_N[j], \hat{B}_N[j], \hat{A}'_N[j], \hat{B}'_N[j] :$ $j > 1$ are chosen uniformly at random. Similarly, for $P_i \in Q$, the values $\hat{C}_i, \hat{C}'_i, \hat{D}_i, \hat{\varepsilon}_i$ are indistinguishable from $C^*_i, C'^*_i, D^*_i, \varepsilon^*_i$. For $P_i \in Q \setminus \{N\}$ this follows because the simulator performs the same steps as in the real protocol, while the semantic security of CF encryption implies that the adversary cannot distinguish the encryptions of zero $\hat{C}_N$, $\hat{C}'_N$ and $\hat{D}_N = [\gamma - \sum_{i \in Q \setminus \{N\}} s_i \sum_{j \in Q \setminus \{N\}} s'_j - \sum_{j \in Q \setminus \{N\}} t_j]_y$ from the real values C^*_N, ${C'_N}^*$ and D^*_N respectively. Now trapdoor τ_N enables the zero knowledge simulator S_2 to compute fake proofs $\hat{\varepsilon}_N$ on the values $\hat{C}_N$, $\hat{C}'_N$ and $\hat{D}_N$, without the adversary noticing.
Finally the simulator participates in distributed CF decryption for the honest players. This is a perfect simulation of the real protocol because it has access to the ideal functionality for CF key share generation (which exists assuming that protocol is secure).

Proof of Proposition 3 It is clear that the set Q is well-defined as it is a function of only publicly available information. We have that for any set $R \subseteq Q$ of T shares, $s_i = \sum_{j \in R} \Lambda_{j,R} \cdot s_{ij}$, $s'_i = \sum_{j \in R} \Lambda_{j,R} \cdot s'_{ij}$ and $t_i = \sum_{j \in r} \Lambda_{j,R} \cdot t_{ij}$ in $\mathbb{Z}_p$. Hence $x = \sum_{i \in Q} s_i = \sum_{i \in Q} (\sum_{j \in R} \Lambda_{j,R} \cdot s_{ij}) = \sum_{j \in R} \Lambda_{j,R} \cdot (\sum_{i \in Q} s_{ij}) = \sum_{j \in R} \Lambda_{j,R} \cdot x_j$, i.e. x can be publicly re-constructed from any set of T shares. By a similar argument x' and b can be publicly re-constructed from any set of T shares.

Proof of Proposition 4 By the correctness of CF decryption, we have $\gamma = \sum_{k=0}^{2(\ell-1)} \gamma_k c_B^k = \sum_{k=0}^{2(\ell-1)} \sum_{i,j \in Q} \sum_{f+g=k} \alpha_{if} \alpha'_{jg} c_B^k + \sum_{i \in Q} \beta_{ik} c_B^k$
$= \sum_{i,j \in Q} \sum_{k=0}^{2(\ell-1)} \sum_{f+g=k} \alpha_{if} \alpha'_{jg} c_B^{f+g} + \sum_{i \in Q} \sum_{k=0}^{2(\ell-1)} \beta_{ik} c_B^k = (\sum_{i \in Q} \sum_{f=0}^{(\ell-1)} \alpha_{if} c_B^f) \cdot$ $(\sum_{j \in Q} \sum_{g=0}^{(\ell-1)} \alpha'_{jg} c_B^g) + \sum_{i \in Q} t_i = (\sum_{i \in Q} s_i) \cdot (\sum_{j \in Q} s'_j) + \sum_{i \in Q} t_i = xx' + b.$

Proof of Theorem 5 There are two simulators, depending on whether an (honest) threshold passes verifications.

Case 1: at least t pass verification. We may assume that all players in $\overline{B}$ commit to values $\{u_i\}_{i\in\overline{B}}$ which are unrelated to the set $\{u_j\}_{j\in B}$ since otherwise $\mathcal{A}$ finds a pair of message vectors $\boldsymbol{m}, \boldsymbol{m}'$ satisfying $\boldsymbol{m}[i] = u_i, \boldsymbol{m}'[j] = u_j$ for which the event $D(s, \boldsymbol{m}, \boldsymbol{m}', z_D) = 1$ occurs with noticeably greater probability in the real protocol than the simulated. By the soundness of $\mathcal{P}_{\mathsf{eq}}$ it holds that $\mathsf{Dec}(\mathsf{sk}_1, C'_i) = \mathsf{Dec}(\mathsf{sk}_2, \overline{C}_i)$ for all $i \neq N$ or the simulation aborts. Thus in the real protocol the values C' and $\overline{C}$ satisfy the relation $C' - C'_N \equiv \overline{C} - \overline{C}_N$. Therefore we need to show that in the simulation P_N produces C'_N and $\overline{C}_N$ which satisfy this and are indistinguishable from the corresponding values produced in the real protocol. This follows from the semantic security of Enc_1 and Enc_2 and the fact that in the simulation C' and $\overline{C}$ are explicitly constructed as the bit-wise sum of $C'_1, \ldots, C'_{N-1}$ and $\overline{C}_1, \ldots, \overline{C}_{N-1}$ which contain identical values. Finally, the soundness of Σ_{CD} implies that the simulation succeeds except with negligible probability.

Case 2: fewer than t pass verification. Then the adversary may learn the output and may not pass it on to the result party. It can choose random blinding factors and use the commitment simulators and ZK simulators to simulate the honest participants perfectly.

Protocol 4

Common Input : Ballots $B_1, \ldots, B_M$ in encrypted preference-order representation.

Private Input : $\mathsf{sk}_i \leftarrow \mathsf{Share}(\mathsf{pk}, \mathsf{sk}, \mathbb{A}_{T\text{-Th}}) : \mathsf{sk}_i = (\alpha_i, \beta_i)$.

Public Output : All intermediate tallies of all candidates until termination with a candidate who wins a majority.

Player P_i :

```
1: $S_\mathrm{C} \leftarrow \{1, \ldots, c\}$, $b_\mathrm{ELECT} \leftarrow$ False, $\lambda \leftarrow 1$
2: while not $b_\mathrm{ELECT}$ do
3:     $\mathbf{v}_\mathrm{TALLY} \leftarrow \underbrace{\mathsf{Enc}^*_\mathsf{tgt}(0), \ldots, \mathsf{Enc}^*_\mathsf{tgt}(0)}_{|S_C|}$        ▷ Tally first preference votes.
4:     for $1 \le n \le M$ do
5:         Let $(\mathbf{v}_1 \ldots, \mathbf{v}_k) \leftarrow B_n|_{S_C}$   ▷ Compute first preference vote for voter $n$.
6:         $\boldsymbol{\pi}_1 \leftarrow \mathsf{Enc}^*_\mathsf{src}(1)$
7:         $\mathbf{v}_\mathrm{FP} \leftarrow \mathbf{v}_1 \boxtimes_\mathsf{src} \boldsymbol{\pi}_1$
8:         for $2 \le j \le \lambda$ do
9:             $\boldsymbol{\pi}'_j \leftarrow \boldsymbol{\pi}_{j-1} \boxtimes_\mathsf{src} (\mathsf{Enc}^*_\mathsf{src}(1) \ominus_\mathsf{src} \Sigma_{S_\mathrm{C}}(\mathbf{v}_{j-1}))$
10:            $\boldsymbol{\pi}_j \leftarrow \mathsf{Switch}_{\mathsf{tgt}\to\mathsf{src}}(\mathsf{pk}, \boldsymbol{\pi}'_j, \mathsf{sk}_i)$
11:            $\mathbf{v}'_j \leftarrow \mathbf{v}_j \boxtimes_\mathsf{src} \boldsymbol{\pi}_j$
12:            $\mathbf{v}_\mathrm{FP} \leftarrow \mathbf{v}_\mathrm{FP} \oplus_\mathsf{tgt} \mathbf{v}'_j$
13:        end for        ▷ Add first preference vote of voter $n$ to running tally.
14:        $\mathbf{v}_\mathrm{TALLY} \leftarrow \mathbf{v}_\mathrm{TALLY} + \mathbf{v}_\mathrm{FP}$
15:    end for
16:    $\lambda \leftarrow \lambda + 1$        ▷ Decrypt running tally.
17:    $(n_1, \ldots, n_c) \leftarrow \mathsf{Dec}_\mathsf{tgt}(\alpha_i, \mathbf{v}_\mathrm{TALLY})$
18:    $j^* \leftarrow \mathrm{argmax}_{j \in S_C}(n_j)$
19:    if $n_{j^*} > \lfloor \frac{M}{2} \rfloor$ or $\lambda > k$ then
20:        $b_\mathrm{ELECT} \leftarrow$ True        ▷ Declare winner.
21:        *Broadcast* $(\mathsf{elect}, c_{j^*})$
22:        return
23:    else
24:        $j^* \leftarrow \mathrm{argmin}_{j \in S_C}(n_j)$        ▷ Eliminate candidate with fewest votes.
25:        *Broadcast* $(\mathsf{eliminate}, c_{j^*})$
26:        $S_\mathrm{C} \leftarrow S_\mathrm{C} \backslash \{j^*\}$
27:    end if
28: end while
```

Fig. 14. Tallying IRV ballots with distributed encryption switching.

Simulator for Protocol 4

Input $(y_{\mathsf{pke}}, z_{\mathsf{pke}}), \{[\gamma_k]_y\}_k$.

- Assume P_N is honest.
- Let B be the set of corrupted players.

1. Let $(\hat{\sigma}_N, \tau_N) \leftarrow S_1(1^\lambda)$.
2. Perform Steps 1-3 on behalf of honest parties.
3. Compute s_i, s'_i and t_i for all $P_i \in Q\backslash\{N\}$. Choose s_{Ni}, s'_{Ni} and t_{Ni} at random in $\mathbb{Z}_p$ for $i \in Q \cap B$. Choose $y' \in_R \mathbb{G}_1, z' \in_R \mathbb{G}_2$.
4. Compute $\{\hat{A}_i, \hat{B}_i, \hat{A}'_i, \hat{B}'_i, \hat{C}_i, \hat{C}'_i, \hat{D}_i, \hat{\varepsilon}_i\}$ as in the real protocol for $i \in Q\backslash\{N\}$. Let $\hat{A}_N[1] = y_{\mathsf{pke}} \cdot (\prod_{i\in Q\backslash\{N\}} A_i[1])^{-1}, \hat{B}_N[1] = y' \cdot (\prod_{i\in Q\backslash\{N\}} \hat{B}_i[1])^{-1}, \hat{A}'_N[1] = z_{\mathsf{pke}} \cdot (\prod_{i\in Q\backslash\{N\}} \hat{A}'_i[1])^{-1}, \hat{B}'_N[1] = z' \cdot (\prod_{i\in Q\backslash\{N\}} \hat{B}'_i[1])^{-1}$. Let $\hat{A}_N[j] = (\hat{A}_N[1]/\prod_{i\in Q\cap B} g_{\mathsf{pke}}^{s_{Ni}\cdot\Lambda_{i,B}})^{\Lambda_{j,B}^{-1}}, \hat{A}'_N[j] = (\hat{A}'_N[1]/\prod_{i\in Q\cap B} h_{\mathsf{pke}}^{s'_{Ni}\cdot\Lambda_{i,B}})^{\Lambda_{j,B}^{-1}}, \hat{B}_N[j] = (\hat{B}_N[1]/\prod_{i\in Q\cap B} g_{\mathsf{pke}}^{t_{Ni}\cdot\Lambda_{i,B}})^{\Lambda_{j,B}^{-1}}, \hat{B}'_N[j] = (\hat{B}'_N[1]/\prod_{i\in Q\cap B} h_{\mathsf{pke}}^{t_{Ni}\cdot\Lambda_{i,B}})^{\Lambda_{j,B}^{-1}}$ for $j > 1$. Let $\hat{C}_N = ([0]_y, \ldots, [0]_y), \hat{C}'_N = ([0]_y, \ldots, [0]_y), \hat{D}_N = (\hat{D}_{N0}, \ldots, \hat{D}_{N(2\ell-2)})$ where $\hat{D}_{Nk} = [\gamma_k]_y - \sum_{i,j\in Q\backslash\{N\}} \sum_{f+g=k} \hat{C}_{if}\hat{C}'_{jg} - \sum_{i\in Q\backslash\{N\}} \hat{D}_{ik}$. Let $\hat{\varepsilon}_N \leftarrow S_2(\tau_N, \hat{C}_N, \hat{C}'_N, \hat{D}_N, \hat{A}_N, \hat{B}_N)$. Broadcast $\{\hat{A}_i, \hat{B}_i, \hat{A}'_i, \hat{B}'_i, \hat{C}_i, \hat{C}'_i, \hat{D}_i, \hat{\varepsilon}_i\}$ for $i \in \overline{B}$.
5. Check that Equation 2 holds and that published proofs are valid on behalf of $P_i \in \overline{B}$. If $(s_{ji}, s'_{ji}, t_{ji})$ does not satisfy this equation for some P_j, broadcast $(P_i, \mathsf{complain}, P_j)$.
6. For every player P_j for which a valid complaint $(P_{i_\alpha}, \mathsf{complain}, P_j)$ was made, the simulator reconstructs $s_j, s'_j, t_j, A_j, A'_j, B_j, B'_j, C_j, C'_j, D_j$ accordingly.
7. The simulator runs distributed CF decryption on behalf of $P_i \in \overline{B}$.

Fig. 15. Simulator for the key generation protocol for one-time homomorphic cryptosystem.

Simulator for π_{SWITCH}
Input : $(\mathsf{pk}_1, c, \overline{c})$, threshold t.

– Let B be the set of corrupted players. Assume $|N \backslash B| > t$.

1. Let $ck \leftarrow \mathcal{K}'(1^\lambda)$. Pass $(\mathsf{setup}, 1^\lambda)$ to $\mathcal{F}_{\mathsf{SUM}}$.
2. Let $(\hat{\sigma}_N, \tau_N) \leftarrow S_1(1^\lambda)$.
3. Perform Steps 1–3 on behalf of honest players $N \backslash B$ except that (send, C'_i) and $(\mathsf{send}, \overline{C}_i)$ are passed to $\mathcal{F}_{\mathsf{SUM}}$.
4. Let $C'_N = \mathsf{Enc}_1(0)$ and $\overline{C}_N = \mathsf{Enc}_2(0)$. Pass (send, C'_N) and $(\mathsf{send}, \overline{C}_N)$ to $\mathcal{F}_{\mathsf{SUM}}$ and let C' and $\overline{C}$ be the output.
5. Perform Steps 4–5 on behalf of the honest players except that ξ_i are simulated, i.e., let $d \leftarrow c \cdot C', d_i \leftarrow d^{\mathsf{sk}_i}, \xi_i \leftarrow \Sigma.\mathsf{sim}(d, d_i, \mathsf{pk}, \mathsf{vk}_i), \hat{m} \leftarrow \prod_{i=1}^{T} d_i$ and $\hat{c} = \mathsf{Enc}_2^*(\hat{m}) \cdot \overline{C}^{-1}$. Output $\perp$ if $\hat{c} \neq \overline{c}$.

Fig. 16. Simulator for protocol π_{SWITCH}.