Universally Verifiable MPC and IRV Ballot Counting

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

[1] Department of Computing and Information Systems, The University of Melbourne, Melbourne, Australia
kramchen@gmail.com, {christopher.culnane,vjteague}@unimelb.edu.au
[2] ICTEAM, UCLouvain, 1348 Louvain-la-Neuve, Belgium
olivier.pereira@uclouvain.be
[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.

Keywords: Multiparty computation · Elections · Voting · Instant runoff voting · Verifiable computation · Verifiability

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

I. Goldberg and T. Moore (Eds.): FC 2019, LNCS 11598, pp. 301–319, 2019.
https://doi.org/10.1007/978-3-030-32101-7_19

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 [1,5,14]. 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 [9] 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 [11] 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 [28].

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 (Fig. 1). It supports threshold key generation in the malicious setting with static corruption.

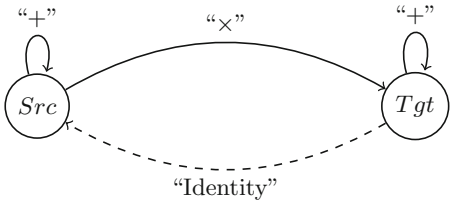

Fig. 1. Operations supported by our encryption scheme. Continuous (resp. dashed) arrows refer to non-interactive (resp. interactive) operations.

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 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.* [13], 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 [11] 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 [8], 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. [3] add universally verifiable proofs of correctness to SPDZ [15], 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 [7], 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 [28] 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. [10], who propose new encryption schemes and switching protocols following [13], 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 [10]. 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. [2] 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 [6,19,20,25–27].

However, these all use mix-nets [23], 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 h. The second, with 6 candidates and 4 elimination rounds, took 15 h. This does not include the proofs of correct switching, which would add a constant multiplicative factor. The details are in Appendix J of the full version of this paper, at https://eprint.iacr.org/2018/246.

1.4 Structure of This Paper

The next section contains cryptographic background. In Sect. 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 Sect. 3, we tackle the problem of constructing a distributed key generation procedure for this protocol. Then in Sect. 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 Sect. 5.

1.5 Background

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

Definition 1 (Access Structure [22,30]). *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 [4,30]). *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 [4] that every every linear secret-sharing scheme (LSSS) satisfies the following property, called *linear-reconstruction* in [30]. 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 Sect. 3, we require the notion of *projecting bilinear group generators* [17]. Our specific choice of generator will be a variant of the polynomial-induced projecting generator introduced by Herold et al. [21], tailored for the asymmetric pairing setting.

Definition 4 (Bilinear Group Generator [17]). *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 [17]). *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 [21]. 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 [16]). *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 [29] 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 [11] showed how to generalise earlier work on 2DNF formulae [8] 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 of the full version. Let $\Pi_{\mathsf{range}} = (G_{\mathsf{range}}, P_{\mathsf{range}}, V_{\mathsf{range}})$ 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 Π_{bit} be the corresponding proof system. Let $\Pi_{\mathsf{eq}} = (G_{\mathsf{eq}}, P_{\mathsf{eq}}, V_{\mathsf{eq}})$ 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 [8]. 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 [17], who shows how to build such schemes from projecting pairings and, more recently by Herold et al. [21] who show how to build them from hidden matrix-rank based indistinguishability assumptions [16] 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 [29].

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 of the full version. Here we construct a projecting bilinear group as a special case 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 of the full version.

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 Sect. 2. Traditional protocols for threshold key generation [18,24] 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 [11], 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 [24] or the later protocol by Gennaro *et al.* [18].

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 [12,18,24], 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 of the full version.

Recall the security properites of a distributed key generation protocol [18].

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 Figs. 2 and 3. The NIZKs are described in Appendix D of the full version.

3.1 Protocol Description and Security Properties

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

Proof. Proofs of this theorem and the following two propositions are in Appendix K of the full version.

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.*

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

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 Fig. 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

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 $1 \leq j \leq N$. 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.

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 of the full version.

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

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

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.

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 [28] and given in Appendix H of the full version. 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 of the full version.

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 of the full version, 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 $\Pi_{\mathsf{src}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_{\mathsf{src}}, \mathsf{Dec}_{\mathsf{src}})$ and $\Pi_{\mathsf{tgt}} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}_{\mathsf{tgt}}, \mathsf{Dec}_{\mathsf{tgt}})$

of Sect. 2. Suppose that $\Pi_{\mathsf{tgt}} \rightarrow \Pi_{\mathsf{src}}$ 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 $\mu_n : \{1, \ldots, k\} \rightarrow \{1, \ldots, c\}$ 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 Fig. 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 $\Sigma_{S_C}(\mathbf{v}_i)$ be the homomorphic sum of the entries of the i^{th} preference vector over uneliminated candidates. Clearly $\Sigma_{S_C}(\mathbf{v}_i)$ 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

$$\boldsymbol{\pi}_j^{(l)} := \bigboxtimes_{\mathsf{src}}{}_{1 \leq j' \leq j} \left(\mathsf{Enc}_1^*(1) - \Sigma_{S_C}(\mathbf{v}_{j'})\right) : j \leq l$$

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.[1] 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 Fig. 14, Appendix J of the full version.

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.[2] 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 Fig. 5. The experiments were performed on an Intel i7-6770HQ with 4 cores (8 threads) and 32 GB 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 0.0671 s, while switching averages 0.0971 s. The code is available at https://github.com/vteague/PPAT/tree/chris-dev.

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

	District	
	Albury (5 candidates)	Auburn (6 candidates)
No. ballots	46347	43738
Ballot encryption time	3069 s	3936 s
No. elimination rounds	1	4
Count time	6979 s	54637 s

[1] For example, the use of a "stop" candidate by [20] remedies the case that a ballot is exhausted prematurely.

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

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.

Acknowledgement. Olivier Pereira is grateful to the Belgian Fund for Scientific Research (F.R.S.- FNRS) for its financial support provided through the SeVoTe project, to the European Union (EU) and the Walloon Region through the FEDER project USERMedia (convention number 501907-379156), and to the Melbourne School of Engineering for its fellowship.

A Appendix

Appendices are in the full version of the paper on the IACR eprint archive at https://eprint.iacr.org/2018/246.

References

1. 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, EVT/WOTE 2009, p. 10. USENIX Association, Berkeley (2009). http://dl.acm.org/citation.cfm?id=1855491.1855501
2. 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)
3. Baum, C., Damgård, I., Orlandi, C.: Publicly auditable secure multi-party computation. In: Abdalla, M., De Prisco, R. (eds.) SCN 2014. LNCS, vol. 8642, pp. 175–196. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10879-7_11. Also Cryptology ePrint Archive, Report 2014/075: http://eprint.iacr.org/2014/075
4. Beimel, A.: Secure schemes for secret sharing and key distribution. Ph.D. thesis, Israel Institute of Technology (1996)
5. Benaloh, J., et al.: Star-vote: a secure, transparent, auditable, and reliable voting system. CoRR abs/1211.1904 (2012). http://arxiv.org/abs/1211.1904
6. 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 (2009). https://doi.org/10.1109/TIFS.2009.2033757
7. Bernhard, D., Cortier, V., Pereira, O., Smyth, B., Warinschi, B.: Adapting helios for provable ballot privacy. In: Atluri, V., Diaz, C. (eds.) ESORICS 2011. LNCS, vol. 6879, pp. 335–354. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23822-2_19
8. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-30576-7_18

9. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. ACM Trans. Comput. Theory **6**(3), 13:1–13:36 (2014)
10. Castagnos, G., Imbert, L., Laguillaumie, F.: Encryption switching protocols revisited: switching modulo p. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 255–287. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_9
11. 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, CCS 2015, pp. 1518–1529. ACM, New York (2015). https://doi.org/10.1145/2810103.2813624. http://doi.acm.org/10.1145/2810103.2813624
12. 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, WPES 2013, pp. 131–142. ACM, New York (2013). https://doi.org/10.1145/2517840.2517852. http://doi.acm.org/10.1145/2517840.2517852
13. Couteau, G., Peters, T., Pointcheval, D.: Encryption switching protocols. In: Robshaw, M., Katz, J. (eds.) CRYPTO 2016, Part I. LNCS, vol. 9814, pp. 308–338. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53018-4_12
14. Cramer, R., Gennaro, R., Schoenmakers, B.: A secure and optimally efficient multi-authority election scheme. In: Fumy, W. (ed.) EUROCRYPT 1997. LNCS, vol. 1233, pp. 103–118. Springer, Heidelberg (1997). https://doi.org/10.1007/3-540-69053-0_9
15. Damgård, I., Pastro, V., Smart, N., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 643–662. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32009-5_38
16. Escala, A., Herold, G., Kiltz, E., Ràfols, C., Villar, J.: An algebraic framework for diffie-hellman assumptions. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 129–147. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40084-1_8
17. Freeman, D.M.: Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 44–61. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13190-5_3
18. Gennaro, R., Jarecki, S., Krawczyk, H., Rabin, T.: Secure distributed key generation for discrete-log based cryptosystems. J. Cryptol. **20**(1), 51–83 (2007). https://doi.org/10.1007/s00145-006-0347-3
19. Goh, E.-J., Golle, P.: Event driven private counters. In: Patrick, A.S., Yung, M. (eds.) FC 2005. LNCS, vol. 3570, pp. 313–327. Springer, Heidelberg (2005). https://doi.org/10.1007/11507840_27
20. Heather, J.: Implementing STV securely in prêt à voter. In: Proceedings of the 20th IEEE Computer Security Foundations Symposium, CSF 2007, pp. 157–169. IEEE Computer Society, Washington (2007). https://doi.org/10.1109/CSF.2007.22
21. 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

22. Ito, M., Saito, A., Nishizeki, T.: Secret sharing scheme realizing general access structure. Electron. Commun. Jpn (Part III: Fundam. Electron. Sci.) **72**(9), 56–64 (1989). https://doi.org/10.1002/ecjc.4430720906. https://onlinelibrary.wiley.com/doi/abs/10.1002/ecjc.4430720906
23. Park, C., Itoh, K., Kurosawa, K.: Efficient anonymous channel and all/nothing election scheme. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 248–259. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-48285-7_21. http://dl.acm.org/citation.cfm?id=188307.188351
24. Pedersen, T.P.: A threshold cryptosystem without a trusted party. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 522–526. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-46416-6_47. http://dl.acm.org/citation.cfm?id=1754868.1754929
25. Ryan, P.Y.A.: Prêt à voter with paillier encryption. Math. Comput. Model. **48**(9–10), 1646–1662 (2008). https://doi.org/10.1016/j.mcm.2008.05.015
26. 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, WITS 2005, pp. 81–88. ACM, New York (2005). https://doi.org/10.1145/1045405.1045414. http://doi.acm.org/10.1145/1045405.1045414
27. 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, EVT/WOTE 2009, p. 13. USENIX Association, Berkeley (2009). http://dl.acm.org/citation.cfm?id=1855491.1855504
28. Schoenmakers, B., Veeningen, M.: Universally verifiable multiparty computation from threshold homomorphic cryptosystems. In: Malkin, T., Kolesnikov, V., Lewko, A.B., Polychronakis, M. (eds.) ACNS 2015. LNCS, vol. 9092, pp. 3–22. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-28166-7_1. http://eprint.iacr.org/2015/058
29. 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
30. Waters, B.: Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19379-8_4. http://dl.acm.org/citation.cfm?id=1964658.1964664