Universally Verifiable MPC with Applications to IRV Ballot Counting

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

[1]Department of Computing and Information Systems, the University of Melbourne
[2]ICTEAM – Université catholique de Louvain

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. We demonstrate the scalability of distribution switching as a viable approach to secure vote tallying by implementing a private verifiable form of Instant Runoff Voting on real Australian election data comprising 40,000 votes.

1 Introduction

We explore the design of efficient universally verifiable MPC protocols, motivated by applications to the counting of complex ballots. *Universal verifiability* means that the computation should be verifiably correct, even to those who do not participate, and even if all parties involved in the computation are misbehaving. This 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.

We also require privacy as long as the number of trustees behaving honestly is above a certain threshold. As trustees must be able to compute the election results (and therefore have access to the votes), and in the absence of setup assumption (anonymous channel, tamper-proof devices, *etc.*), this appears to be the best we can hope for.

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 [ADPQ09] [BBK$^+$12]. We are interested in complex election schemes in which more than a simple sum is needed. Our particular application is Instant Runoff Voting (IRV), in which candidates are progressively eliminated until one has the majority of votes. For this case, levelled homomorphic encryption 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 would be the total number of candidates (minus 2), which might be a lot more than the actual number of eliminations.

An alternative approach is to add universally verifiable proofs of correctness to an information-theoretic MPC protocol [BDO14]. These naturally scale to arbitrary multiplications, with cost proportional to the total number actually done. However, the structure of these 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.

Recently Catalano and Fiore [CF15] showed how to generalise earlier work on 2DNF formulae [BGN05] 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.

1.1 Summary of our contribution

We design a very simple universally verifiable MPC protocol based on two components.

1. *A somewhat homomorphic encryption scheme with threshold key generation in the malicious static adversary setting.* It is similar to [CF15] in allowing arbitrary additions in a source space, then one multiplication. However, additions in the target space can be performed without expansion. Our threshold key generation protocol allows efficient proofs of correct decryption.

2. *A multiparty switching protocol that transforms a ciphertext from the target space, that is, resulting from a homomorphic multiplication, into a ciphertext in the source space, hence making it possible to perform more multiplications if needed.* We show that this protocol is universally verifiable in the setting of [SV15].

Since our scheme is additively homomorphic in the source and target spaces, we have enough to perform arbitrary computation. Every step is universally verifiable. Our scheme only requires computation in prime order groups using standard assumptions.

1.2 Comparison with related work on MPC

Our approach bears some resemblance with the encryption-switching approach of Couteau *et al.* [CPP16], 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.

Boneh, Goh and Nissim [BGN05] consider evaluation of 2DNF formulas on ciphertexts. The paper mentions universally verifiable computation as an application, including in the election context. Our paper essentially solves the question of making this efficient in a multi-authority setting, by offering efficient threshold key operations. BGN assumes a single authority, or a trusted dealer with secure erasure, or other more complicated solutions.

Catalano and Fiore [CF15] 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, nor is it clear how to remove the requirement of non-collusion by servers.

Damgård et al. [DPSZ11] show how to bootstrap somewhat homomorphic encryption to achieve pure multiparty computation via pre-processing. The focus of their scheme is on batch/SIMD evaluation of inputs while minimising computational overhead and relies on non-standard zero knowledge proofs. While our (online) protocols could in any case possibly utilise such amortisation techniques, for example via pre-computation of fixed-based exponentiations, we show in Appendix G that our scheme is already competitive in terms of total arithmetic operations performed during the lifetime of the scheme. Additionally our evaluation algorithm supports malicious case secure threshold key generation and uses only standard sigma protocols.

Three recent works address universally verifiable MPC. Their main bottleneck is key generation. Baum et al. [BDO14] rely on SPDZ which uses a somewhat homomorphic encryption scheme that has n-out-of-n key generation in the covert adversary model, and therefore only offers confidentiality in that model. We would be interested in having security in the traditional malicious adversary setting.

Schoenmakers and Veeningen [SV15] rely on Damgaard-Jurik encryption, which supports efficient threshold key generation, if an RSA modulus with unknown factorization is available, bringing us back to the same difficulties of [BGN05].

We address these issues and offer an efficient solution that is compatible with the settings of both papers.

The most closely related work is [CIL17], with new encryption schemes and switching protocols following [CPP16], but working in prime order groups (like we do), hence also supporting threshold operations. They still combine additively and multiplicatively homomorphic schemes (while we use a somewhat homomorphic approach). The main downside of their work is that they rely on the hardness of DDH in very specific groups: "subgroups of the class group of an order of a quadratic field of discriminant $-p^3$". They also need to work in subgroups of unknown order, which makes ZK proofs more expensive.

Our protocol works in a more standard computational setting, with efficiency and compatibility advantages.

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 [CIL17]. However, a computation with unbounded successive multiplicative homomorphic operations would be faster with their method.

1.3 Counting IRV Ballots

In *Instant runoff voting (IRV)*, each voter lists 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.

Since the number of possible votes is more than $c!$ (where c is the number of candidates), this may be much larger than the number of votes actually cast. This introduces the possibility of coercion by 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. Heather [Hea07] describes how

to modify Prêt à Voter [Rya05] with re-encryption mixes to accommodate Single Transferable vote via lazy decryption semantics. Goh and Golle [GG05] describe a primitive called an event-driven private counter which allows private non-interactive tallying of ballots in a ranked voting scheme. Ryan [Rya08] describe a method for modifying Prêt à Voter to accommodate ranked voting, however as noted in [RT09] this construction is not receipt-free across the full space of possible permutations. Benaloh *et al.* [BMN+09] describe a protocol for private tallying of Single Transferable votes using interactive mixes to convert ballots between different representations amenable to tallying.

However, these all use mix-nets [PIK94] during the private tallying phase, with a number of schemes additionally utilising mix-net like techniques for secure ballot construction. Despite their increasing usage, mix-net based tallying schemes remain heavy to deploy due to their sequential behavior, and 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.

1.3.1 Implementation

We implemented the single-authority version of our cryptosystem and switching protocol and used it to count two real IRV elections 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 factor. The details are in Section 7.1.1.

1.4 Detail of our contributions

We construct secure N-party encryption switching between any source and target cryptosystems with some minimal homomorphic properties; in essence these are that encryption is additively homomorphic on source and target plaintext spaces, and a ring structure exists on both of these spaces. Our construction builds upon the mask-decrypt-unmask paradigm introduced by [CPP16], specifically a blinding factor used to homomorphically blind the source plaintext, is in parallel encrypted under the target scheme. Next, the players combine decryption shares of the source ciphertext, recovering the blinded plaintext, which can then be re-encrypted in the target scheme.

We present a new candidate cryptosystem with which to instantiate source and destination encryption schemes for the N-party encryption switching primitive. This scheme is based upon groups with asymmetric pairings, for which the DDH problem is hard in each group. In addition to achieving public verifiability, an advantage of using pairings in this setting is that one-direction of encryption switching is essentially for free. While one might imagine using an existing pairing-based homomorphic cryptosystem here, e.g. [BGN05], the lack of a practical distributed key generation protocol for such cryptosystems flies in the face of the necessity for ab initio calculations vital to ensuring the fairness and verifiability of the distributed platform upon which secure vote-tallying will run. Therefore the construction of a pairing based homomorphic cryptosystem on *prime-order* groups, for which a secure and robust key generation procedure can be derived, is a central thrust of this work. To this purpose we propose the use of cryptosystems derived from projecting pairings [GS08, Fre10]. As shown by [Fre10], projecting pairings facilitate the construction of sub-group indistinguishability based cryptosystems comparable

to [BGN05]. Such cryptosystems are readily constructed from vanilla pairings on prime-order source and target groups. Herold *et al.* [HHH+14] show that such projecting pairings are readily obtained from hidden matrix-rank based indistinguishability assumptions [EHK+13] on the source group of symmetric pairings. The corresponding indistiguishability assumption increases in difficulty, at least in the generic group model, as the rank increases [EHK+13]. We show that this construction naturally extends to asymmetric pairings. In fact, the underlying indistinguishability problem induced by this pairing on both source groups induces a generalisation of the well-known XDH problem [Sco02].

Next we tackle the problem of constructing a distributed key generation procedure for this protocol. While distributed key generation protocols for general public key-encryption are well-established [Ped91, GJKR07, CGGI13], a critical obstacle to their adoption in our setting is the requirement of robust re-construction of *powers of a secret*. Specifically our sub-group based cryptosystem requires exponentiation of group elements by powers of a secret x, while all key-generation protocols in the prior art [Ped91, GJKR07, CGGI13] facilitate only re-construction of linear functions of x, permitted by verifiable secret sharing [Fel87]. A natural solution to this problem would be the use of a multiplicatively homomorphic linear secret sharing scheme [CB87]. On the other hand [CB87] shows that simple candidates for such a primitive fail. The solution turns out to be to view the construction of the square of a secret x as a private quadratic formula, in which the pieces of x are shared across all qualified parties derived by the key-generation protocol. Specifically, we aim to construct a blinded version, i.e, x^2+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 propose to use the Catalano-Fiore transformation [CF15], which enables depth-one multiplications on any linearly homomorphic cryptosystem, such as El Gamal. 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 x and x^2 are derived is much larger. We are able to solve this problem via 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 quite 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 x^2 in the construction of $x^2 + b$. 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 x^2, 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 [Ped91] or the later protocol by Gennaro *et al.* [GJKR07].

2 Background

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

Definition 1 (Access Structure [Wat11])**.** *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, and the sets not in $\mathbb{A}$ are called the unauthorised sets.*

Definition 2 (Linear Secret-Sharing Scheme [Wat11])**.** *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 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 [Bei96] that every every linear secret-sharing scheme (LSSS) as defined above satisfies the following property, called *linear-reconstruction* in [Wat11]. 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.

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 $d = T$.

2.1 Non-Interactive Zero Knowledge

We recall the notion of a non-interactive zero knowledge proof system [BFM88]. Standard techniques can be used to make such proofs non-malleable where necessary [DSDCO+01].

Definition 3 (Non-Interactive Zero Knowledge Proof [Gro10])**.** *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*

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

2.2 Encryption Switching

In this section we define an N-party extension of the encryption switching protocols by Couteau *et al.* [CPP16] as well as a definition of security following the simulation-based paradigm introduced there-in. We will need the notion of twin-ciphertext pair [CPP16] which is augmented with appropriate homomorphic properties on the respective ciphertext spaces.

Definition 4 (Twin-Ciphertext Pair [CPP16])**.** *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 5 (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 input 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 6 (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* $i = 1, 2$*,*

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

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: α

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: α

Figure 1: Experiments used in distributed encryption switching.

Definition 7 (Security of Distributed Encryption Switching)**.** *Figure 1 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.*

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.

2.3 Pairings on Prime-Order Groups

To build our one-time homomorphic cryptosystem of Section 3, we require the notion of *projecting bilinear group generators* [Fre10]. Our specific choice of generator will be a variant of the polynomial-induced projecting generator introduced by Herold et al. [HHH+14], tailored for the asymmetric pairing setting.

Definition 8 (Bilinear Group Generator [Fre10])**.** *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 $\mathcal{P}$ is prime-order if G, G_1, H, H_1, G_t all have prime order p. Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_t, \hat{e}) \leftarrow \mathcal{P}(1^\lambda)$.

Definition 9 (Projecting Bilinear Group Generator [Fre10])**.** *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 and 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 [HHH+14]. 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 10 (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* $G = \mathbb{G}_1^{l+1}, H = \mathbb{G}_2^{l+1}$ *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* $R(\vec{y}, i)$ *to be the vector* $\vec{y}$ *cyclically shifted right by* i *positions. Let* $\vec{a} \otimes \vec{b}$ *be the tensor product of* $\vec{a}$ *and* $\vec{b}$.

1. *Let* $\vec{x}_1 = (-s, 1, 0, \ldots, 0)$,
   $\vec{x}_2 = R(\vec{x}_1, 1), \ldots,$
   $\vec{x}_l = R(\vec{x}_1, l-1)$ *and*
   $\vec{x}'_1 = (-s', 1, 0, \ldots, 0)$,
   $\vec{x}'_2 = R(\vec{x}'_1, 1), \ldots,$
   $\vec{x}'_l = R(\vec{x}'_1, l-1)$ *in* $\mathbb{Z}_p^{l+1}$ *and*
   $\vec{y_1} = \vec{x}_1 \otimes \vec{x}'_1, \ldots,$
   $\vec{y}_{l(i-1)+j} = \vec{x}_i \otimes \vec{x}'_j, \ldots,$
   $\vec{y_{l^2}} = \vec{x}_l \otimes \vec{x}'_l$ *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* $\{g^{\vec{x}_1}, \ldots, g^{\vec{x}_l}\}$ *and* H_1 *be the subgroup of* H *generated by* $\{h^{\vec{x}'_1}, \ldots, h^{\vec{x}'_l}\}$. *Let* G'_t *be the subgroup of* G_t *generated by* $\{\hat{e}(g,h)^{\vec{y}_1}, \ldots, \hat{e}(g,h)^{\vec{y}_{l^2}}\}$.

3. *Define* $e : G \times H \to G_t$ *by*

$$\begin{aligned} e(g^{\vec{a}}, h^{\vec{b}}) =: \hat{e}(g,h)^{\vec{a} \otimes \vec{b}} \\ = (\hat{e}(g^{a_0}, h^{b_0}), \ldots, \\ \hat{e}(g^{a_i}, h^{b_j}), \ldots, \\ \hat{e}(g^{a_l}, h^{b_l})) \end{aligned}$$

   where $\vec{a} = (a_0, \ldots, a_l)$ *and* $\vec{b} = (b_0, \ldots, b_l)$.

4. *For* $\mathbf{g}_1 \in G, \mathbf{h} \in H$ *and* $\mathbf{g}_t \in G_t$, *define*

$$\pi_1(\mathbf{g}) = \mathbf{g}^{(1, s, \ldots, s^l)^T} \quad \pi_2(\mathbf{h}) = \mathbf{h}^{(1, s', \ldots, s'^l)^T}$$
$$\pi_t(\mathbf{g}_t) = \mathbf{g}_t^{((s^i s'^j)_{i,j=0}^l)^T}$$

   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).

Definition 11 (l-Symmetric Cascade Assumption [EHK+13])**.** *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 for all PPT adversaries $\mathcal{A}$*, the difference:*

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

is a negligible function of λ*.*

Definition 12 (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 [Sco02] holds with respect to group ensembles* $\mathcal{D}_1, \mathcal{D}_2$*, iff the External* 1*-Symmetric Cascade Assumption holds with respect to* $\mathcal{D}_1, \mathcal{D}_2$*.*

2.4 CF Encryption

We recall the cryptosystem of Catalano and Fiore [CF15] which is in fact a black-box transformation of virtually any linearly homomorphic cryptosystem into one that supports evaluation of degree two formulae. 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 $(\mathsf{pk}, \mathsf{sk}) \leftarrow \overline{\mathsf{pk}}$.

$\mathsf{Enc}(\mathsf{pk}, M)$ 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 $M = M' + bb'$.

We use non-interactive zero knowledge proofs of the following NP relations. Efficient constructions of these can be found in Appendix B. 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 $\Pi_{\mathsf{mul}} = (G_{\mathsf{mul}}, P_{\mathsf{mul}}, V_{\mathsf{mul}})$ be a non-interactive zero knowledge proof system for the relation $\mathcal{R}_{\mathsf{mul}} = \{(c, \bar{c}, c', \mathsf{pk}_1, \mathsf{pk}_2) | c, c' \in \mathcal{C}_1 \wedge \exists R \in \mathcal{M}_1 \cap \mathcal{M}_2, \overline{r}, r' : \bar{c} = \mathsf{Enc}_2(\mathsf{pk}_2, R; \overline{r}) \wedge c' = \mathsf{Rand}_1(\mathsf{pk}_1, R \otimes_1 c; r')\}$. For $1 \le j \le N$ let σ_j be the common reference string belonging to P_j.

3 Reusable Commitments

Definition 13 (Commitment Scheme [DG03])**.** *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 14 (Non-malleable Commitment Scheme [DG03])**.** *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.*

$$\begin{gathered}\{D(s, \vec{m}, \vec{m}', z_D) : ck \leftarrow \mathcal{K}(1^\lambda), (s, \vec{m}) \leftarrow M(ck, z_M);\\ (\vec{c}, \vec{d}) \leftarrow \mathsf{commit}_{ck}(\vec{m}); \vec{c}' \leftarrow \mathcal{A}(ck, \vec{c}, M, z_M); \vec{d}' \leftarrow \mathcal{A}(\vec{d});\\ \vec{m}' \leftarrow \mathsf{decommit}_{ck}(\vec{c}', \vec{d}')\}_{\lambda, z_M, z_D}\\ \approx_c\\ \{D(s, \vec{m}, \vec{m}', z_D) : (ck, s_{ck}) \leftarrow \mathcal{K}'(1^\lambda); (s, \vec{m}) \leftarrow M(ck, z_M);\\ \vec{m}' \leftarrow \mathcal{S}(ck, s_{ck}, t, M, z_M)\}_{\lambda, z_M, z_D}\end{gathered}$$

4 One-time Multiplicatively Homomorphic Cryptosystem

In this section we describe a homomorphic cryptosystem which supports arbitrarily many additions on the message space, followed by one multiplication, followed by arbitrarily many additions. The basic idea is to combine Freeman's template for generically converting pairing based cryptosystems with composite order message space to prime order message space [Fre10] with our tensor product-based projecting pairing. For efficiency reasons, we instantiate our constructions with the External 1-Symmetric Cascade Assumption, but it is immediate to generalize this to the l-Symmetric variant.

$\mathsf{Setup}(1^\lambda)$: 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 projecting bilinear generator of Def. 10. 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 $\mathbb{G}_1^2, \mathbb{G}_2^2$ and $\mathbb{G}_t^4$ be descriptions of G, H and G_t respectively, and $\{g^{(-s,1)}\}$ and $\{h^{(-s',1)}\}$ 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 2.3.

$\mathsf{Enc}_{\mathsf{src}}(\mathsf{pk}, M)$: Choose a, b at random in $\mathbb{Z}_p$. Let $\mathbf{g}_1 = (g^{\vec{x}})^a = (g^{-as}, g^a)$ and $\mathbf{h}_1 = (h^{\vec{x}'})^b = (h^{-bs'}, h^b)$. 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 $\mathbf{g}_1 = (g^{\vec{x}})^a = (g^{-as}, g^a)$ and $\mathbf{h}_1 = (h^{\vec{x}'})^b = (h^{-bs'}, h^b)$. 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.

$\mathsf{Multiply}_{\mathsf{src}}(\mathsf{pk}, C, C')$: 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.

$\mathsf{Add}_{\mathsf{src}}(\mathsf{pk}, C, C')$: 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 $C'' = (C''_0, C''_1)$.

$\mathsf{Add}_{\mathsf{tgt}}(\mathsf{pk}, C, C')$: 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''.

$\mathsf{Dec}_{\mathsf{src}}(\mathsf{sk}, C)$: 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.

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

Lemma 2. *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 A. □

5 Distributed Key Generation Protocol for One-time Homomorphic Cryptosystem

In this section we describe a key generation protocol for the multiplicatively homomorphic cryptosystem of Section 4. Traditional protocols for threshold key generation [Ped91, GJKR07] 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.

In order to overcome this difficulty, the idea of our protocol is for each party in the qualified set to split their individual secrets into chunks over a small interval for which decryption in the CF cryptosystem is feasible. By a suitable computation on encrypted chunks of the individual secrets of the parties in the qualified set, the product the secrets s and s' can be computed as a list of encrypted chunks. To obscure the recovery of the square in totality, we introduce a blinding factor, which is added to the square prior to decryption. This blinding factor is produced in analogous fashion to the secret itself, namely it is contributed to by all parties in the qualified set and can be recovered by a threshold number of them. Thus, after CF decryption, a blinding of the square of the secret is revealed as a field element in the clear, while the blinding factor is a distributed secret. By the linearity of exponentiation on group elements, the application of 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, this is explained in more detail in the proof of Theorem 3. Like the key generation protocols of [Ped91, CGGI13, GJKR07], 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 15, Appendix C.

We recall the definitions of correctness, resilience and security of a distributed key generation protocol [GJKR07].

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.

Theorem 3. *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.

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 4 and 5 below. Any qualified set $R \subseteq Q$ may compute $\pi_t(z_1, z_2, z_3, z_4)$ as follows

$$\begin{aligned}
\pi_t(z_1,z_2,z_3,x_4) = z_1 z_2^x z_3^{x'} z_4^{xx'} &= \frac{z_1 z_2^x z_3^{x'} z_4^{xx'+b}}{z_4^b} \\
&= \frac{z_1 z_2^x z_3^{x'} z_4^{\gamma}}{z_4^b} \\
&= \frac{z_1 z_2^{\sum_{j\in R} \Lambda_{j,R}(\sum_{i\in Q} s_{ij})} z_3^{\sum_{j\in R} \Lambda_{j,R}(\sum_{i\in Q} s'_{ij})} z_4^{\gamma}}{z_4^{\sum_{j\in R} \Lambda_{j,R}(\sum_{i\in Q} t_{ij})}} \\
&= \frac{z_1 z_2^{\sum_{j\in R} \Lambda_{j,R} x_j} z_3^{\sum_{j\in R} \Lambda_{j,R} x'_j} z_4^{\gamma}}{z_4^{\sum_{j\in R} \Lambda_{j,R} b_j}}
\end{aligned}$$

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.

Protocol 1– Part 1

Common Input : CF public key y. Generators $g_{\mathsf{vss}}, g_{\mathsf{pke}} \in \mathbb{G}_1$ and $h_{\mathsf{pke}} \in \mathbb{G}_2$.

Private Input : P_i holds a share k_i of secret key k, corresponding to public key y.

1. Let $s_i = \sum_{k=0}^{\ell-1} \alpha_{ik} c_B^k, s'_i = \sum_{k=0}^{\ell-1} \alpha'_{ik} c_B^k, t_i = \sum_{k=0}^{2\ell-2} \beta_{ik} c_B^k$ 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 $\vec{v}_i = (s_i, r_{i2}, \ldots, r_{iT}), \vec{v}'_i = (s'_i, r'_{i2}, \ldots, r'_{iT}), \vec{w}_i = (t_i, r''_{i2}, \ldots, r''_{iT})$. P_i computes the share vectors $\vec{s}_i = M\vec{v}_i, \vec{s}'_i = M\vec{v}'_i$ and $\vec{t}_i = M\vec{w}_i$. Let $V_i = g_{\mathsf{vss}}^{\vec{v}_i}, V'_i = g_{\mathsf{vss}}^{\vec{v}'_i}, W_i = g_{\mathsf{vss}}^{\vec{w}_i}$. P_i broadcasts the values $\{V_i, V'_i, W_i\}$. P_i sends $s_{ij} = \vec{s}_i[j], s'_{ij} = \vec{s}'_i[j], t_{ij} = \vec{t}_i[j]$ 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.
4. Let $A_i = g_{\mathsf{pke}}^{\vec{v}_i}, B_i = g_{\mathsf{pke}}^{\vec{w}_i}, A'_i = h_{\mathsf{pke}}^{\vec{v}'_i}, B'_i = h_{\mathsf{pke}}^{\vec{w}_i}$, $C_i = ([\alpha_{i0}]_y, \ldots, [\alpha_{i(\ell-1)}]_y)$, $C'_i = ([\alpha'_{i0}]_y, \ldots, [\alpha'_{i(\ell-1)}]_y)$, $D_i = ([\beta_{i0}]_y, \ldots, [\beta_{i(2(\ell-1))}]_y)$ where $\vec{v}_i, \vec{v}'_i, \vec{w}_i$ are sampled as in Step 1. Let $\varepsilon_i \leftarrow (\mathcal{P}_{\mathsf{range}}((C_{ik})_k, c_A), \mathcal{P}_{\mathsf{range}}((C'_{ik})_k, c_A), \mathcal{P}_{\mathsf{range}}((D_{ik})_k, c_B), \mathcal{P}_{\mathsf{eq}}(A_i[1], \prod_{k=0}^{\ell-1} [\alpha_{ik}]_y^{c_B^k}), \mathcal{P}_{\mathsf{eq}}(A'_i[1], \prod_{k=0}^{\ell-1} [\alpha'_{ik}]_y^{c_B^k}), \mathcal{P}_{\mathsf{eq}}(B_i[1], \prod_{k=0}^{2(\ell-1)} [\beta_{ik}]_y^{c_B^k}))$. P_i broadcasts the values $\{A_i, A'_i, B_i, B'_i, C_i, C'_i, D_i, \varepsilon_i\}$. Note that

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

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 satisfying $V_{\mathsf{range}}(\sigma_j, (C_j, C'_j, D_j), \varepsilon_j) \neq 1 \vee V_{\mathsf{eq}}(\sigma_j, (A_j, A'_j, B_j), (C_j, C'_j, D_j), \varepsilon_j) \neq 1$ the other players in Q reconstruct the values $s_j, t_j, \vec{v}_j, \vec{w}_j, A_j, A'_j, B_j, B'_j, C_j, C'_j, D_j$.
7. For $0 \le k \le 2(\ell-1)$, 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 $\gamma_k \leftarrow \mathsf{Dec}(k_i, \mathsf{ct}_k)$. Outputs $\gamma = \sum_{k=0}^{2(\ell-1)} \gamma_k c_B^k$.

Figure 2: Key gen protocol for one-time homomorphic cryptosystem.

Protocol 1– Part 2

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.

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

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 $y_{\mathsf{pke}}, z_{\mathsf{pke}}, y', z'$. 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 encryption 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). □

Proposition 4. *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. It is clear that the set Q is well-defined as it is a function of only publically available

Simulator for Protocol 1
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 $\hat{D}_{Nk} = [\gamma_k]_y - \sum_{i,j\in Q\backslash\{N\}} \sum_{f+g=k} C_{if}C'_{jg} - \sum_{i\in Q\backslash\{N\}} D_{ik}$. 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 $P_i \in \overline{B}$.

Figure 4: Simulator for the key generation protocol for one-time homomorphic cryptosystem.

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 publically re-constructed from any set of T shares. By a similar argument x' and b can be publically re-constructed from any set of T shares. □

Proposition 5. *The values γ, x, x' and b computed in Step 6 satisfy the relation $\gamma = xx' + b$.*

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

6 Distributed Encryption Switching

In this section we describe a protocol for universally verifiable switching between source and target encryption schemes using only the additive homomorphism on the ciphertext spaces. The protocol and simulator are in Figures 5 and 6. The essential idea is to for each party to contribute an equivalent encryption of a blinding factor under both cryptosystems together

with a zero knowledge proof of correctness. In the source space the blinding factors are homomorphically added to the input ciphertext and the result decrypted under a robust threshold decryption scheme. To this plaintext, the blinding factors under the target encryption scheme are homomorphically subtracted, the result is an encryption of the input message under the target cryptosystem.

In order to blind the ciphertexts without increasing the size of the messages to be decrypted (remember that it requires a DL extraction), we propose to apply the blinding using a xor-sum, rather than the more natural operations over the integers. 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 C. This protocol requires a number of decryption operations that is proportional to the size of the messages to be blinded.

If the ciphertexts are known to be small, the xor-sum can be avoided and we can simply rely on the additive homomorphism of the encryption scheme, like we did for key generation. This provides statistical blinding, and comes with the benefit of being a completely non interactive process.

Our definition of universally verifiable secure computation is derived from [SV15] and given in Appendix F. It formalises the idea that either a threshold of honest participants produces a true answer, or the output fails verification.

Protocol π_{SWITCH}

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

Player P_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 $\perp$.

Figure 5: Protocol π_{SWITCH}.

Theorem 6. *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.*

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 $\hat{c} \neq \overline{c}$.

Figure 6: Simulator for protocol π_{SWITCH}.

Proof. 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 $\vec{m}, \vec{m}'$ satisfying $\vec{m}[i] = u_i, \vec{m}'[j] = u_j$ for which the event $D(s, \vec{m}, \vec{m}', z_D) = 1$ 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. □

Given that the switch is the only operation of our protocols that requires the use 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.

7 Tallying IRV Ballots

In this section we describe how to use the primitives described in the earlier part of this work, to construct a secure distributed protocol for tallying encrypted ballots according to the instant run-off algorithm. Given that ballots are input to the tallying protocol in encrypted

form and that we only reveal the computed tallies of first preference votes after each round of the IRV algorithm, the main challenge is to ensure that the privacy of ballots is maintained between tallying rounds. We will use distributed encrypted 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 Section 3. Suppose that $\Pi_{\mathsf{tgt}} \to \Pi_{\mathsf{src}}$ is a distributed encryption switching protocol, where $\mathsf{Enc}_{\mathsf{src}}$ is used to encrypt plaintext votes. In that case 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 product computations are again possible. In this way, for every round of tallying after the first, distributed encryption switching can be used to ensure that the computation can be used to determine the first preference vote for a continuing candidate on each ballot.

7.1 Protocol Details

Ballot representation. Assume c candidates and M voters. Assume that 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 assume 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 c-tuples of ciphertexts of length k, i.e., $B_n = (\mathbf{v}_1, \ldots, \mathbf{v}_k)$. Figure 9 shows the representation we employ for tallying.

Updating of ballots 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. The ballot representation described in the previous paragraph permits a particularly convenient method for this purpose, while still allowing further preferences to be counted. Specifically, a candidate may be eliminated from ballot B_n simply by striking out the corresponding column in the matrix of preferences. Since each elimination is made as a function of publically verifiable totals, there is no ambiguity as to the representation of any ballot at any stage of tallying. An important feature of this representation is that the sequence of accesses ever made by Protocol 2 is derivable from the sequence of intermediate tallies it produces until termination. Input obliviousness follows. Figure 10 shows a preference-order ballot after a candidate has been eliminated.

Tallying first-preference votes Let $B_n = (\mathbf{v}_1, \ldots, \mathbf{v}_k)$ be a ballot. Let S_{C} be the set of continuing candidates. Define $\Sigma_{S_{\mathrm{C}}}(\mathbf{v}_i)$ to be the homomorphic sum of the entries of the i^{th} preference vector over continuing candidates. Clearly $\Sigma_{S_{\mathrm{C}}}(\mathbf{v}_i)$ is an encryption of 1 iff the i^{th} preference is for a continuing 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 $M' = \mathsf{Dec}_{\mathsf{src}}(\mathsf{sk}, C')$. We have that, after $l \leq k$ rounds of tallying the encrypted 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 a continuing candidate, and an encryption of 1 otherwise. By assumption, 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

[1] For example, the use of a "stop" candidate by [Hea07] remedies the case that a ballot is exhausted prematurely.

of tallying, the homomorphic dot product $\sum_{j=1}^{l} \mathbf{v}_j \boxtimes_{\mathsf{src}} \boldsymbol{\pi}_j$ is simply an encryption of the first preference for a continuing candidate, in indicator format. The protocol is shown in Figure 7.

Theorem 7. *Let ϵ_{IS} be the maximal distinguishing advantage of adversary $\mathcal{A}$ in the ideal switching experiment. Then Protocol 2 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 8. 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 8, 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 $\epsilon \leq \sum_{l=1}^{k} |\Pr[\mathcal{A}(H_1^{(l-1)}) = 1] - \Pr[\mathcal{A}(H_1^{(l)}) = 1]| \leq \sum_{l=1}^{k} \sum_{I=1}^{l} |\Pr[\mathcal{A}(H_{I-1}^{(l)}) = 1] - \Pr[\mathcal{A}(H_I^{(l)}) = 1]| \leq \binom{k}{2} \cdot \epsilon_{\mathsf{IS}}$. □

Claim 8. *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 $\mathcal{O}_{\overline{B}}$. □

7.1.1 Implementation

A proof-of-concept implementation of the IRV counting protocol was made in Python 2.7 using the PPAT library[2] for group operations, based on a BN curve [BN06] with a prime modulus of 256 bits. Ballots were represented as per Figure 9. The implementation did not include the construction or checking of proofs, and ran as a single party. It would be straightforward to include multiple parties with the addition of the appropriate communication code.

The implementation was tested using elections data for the districts of Albury and Auburn for the New South Wales Legislative Assembly election in 2015.[3] The implementation encrypted each of the entries in the ballot matrix prior to commencing the count, to simulate the receipt of encrypted ballots. To improve performance both the parts of the code for ballot encryption, and the running of the protocol in Figure 7, were made to run using multiple threads.

The experiments were performed on an Intel i7-6770HQ with 4 cores (8 threads) and 32GB RAM. The results are shown in Table 1, with timings given in seconds. 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.

8 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 multipli-

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

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

Protocol 2

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 :

$S_{\mathrm{C}} \leftarrow \{1, \ldots, c\}$, $b_{\mathrm{ELECT}} \leftarrow$ False, $\lambda \leftarrow 1$
while not b_{ELECT} **do**
  $\mathbf{v}_{\mathrm{TALLY}} \leftarrow \underbrace{\mathsf{Enc}^*_{\mathsf{tgt}}(0), \ldots, \mathsf{Enc}^*_{\mathsf{tgt}}(0)}_{|S_C|}$ ▷ Tally first preference votes.
  for $1 \leq n \leq M$ **do**
    Let $(\mathbf{v}_1 \ldots, \mathbf{v}_k) \leftarrow B_n|_{S_C}$ ▷ Compute first preference vote for voter n.
    $\boldsymbol{\pi}_1 \leftarrow \mathsf{Enc}^*_{\mathsf{src}}(1)$
    $\mathbf{v}_{\mathrm{FP}} \leftarrow \mathbf{v}_1 \boxtimes_{\mathsf{src}} \boldsymbol{\pi}_1$
    for $2 \leq j \leq \lambda$ **do**
      $\boldsymbol{\pi}'_j \leftarrow \boldsymbol{\pi}_{j-1} \boxtimes_{\mathsf{src}} (\mathsf{Enc}^*_{\mathsf{src}}(1) \ominus_{\mathsf{src}} \Sigma_{S_{\mathrm{C}}}(\mathbf{v}_{j-1}))$
      $\boldsymbol{\pi}_j \leftarrow \mathsf{Switch}_{\mathsf{tgt}\to\mathsf{src}}(\mathsf{pk}, \boldsymbol{\pi}'_j, \mathsf{sk}_i)$
      $\mathbf{v}'_j \leftarrow \mathbf{v}_j \boxtimes_{\mathsf{src}} \boldsymbol{\pi}_j$
      $\mathbf{v}_{\mathrm{FP}} \leftarrow \mathbf{v}_{\mathrm{FP}} \oplus_{\mathsf{tgt}} \mathbf{v}'_j$
    end for ▷ Add first preference vote of voter n to running tally.
    $\mathbf{v}_{\mathrm{TALLY}} \leftarrow \mathbf{v}_{\mathrm{TALLY}} + \mathbf{v}_{\mathrm{FP}}$
  end for
  $\lambda \leftarrow \lambda + 1$ ▷ Decrypt running tally.
  $(n_1, \ldots, n_c) \leftarrow \mathsf{Dec}_{\mathsf{tgt}}(\alpha_i, \mathbf{v}_{\mathrm{TALLY}})$
  $j^* \leftarrow \mathrm{argmax}_{j \in S_C}(n_j)$
  if $n_{j^*} > \lfloor \frac{M}{2} \rfloor$ **or** $\lambda > k$ **then**
    $b_{\mathrm{ELECT}} \leftarrow$ True ▷ Declare winner.
    Broadcast $(\mathsf{elect}, c_{j^*})$
    return
  else
    $j^* \leftarrow \mathrm{argmin}_{j \in S_C}(n_j)$ ▷ Eliminate candidate with fewest votes.
    Broadcast $(\mathsf{eliminate}, c_{j^*})$
    $S_{\mathrm{C}} \leftarrow S_{\mathrm{C}} \backslash \{j^*\}$
  end if
end while

Figure 7: Tallying IRV ballots with distributed encryption switching.

Simulator for Protocol 2
Input : All intermediate tallies until termination and products $\boldsymbol{\pi}_j^{(l)}, \boldsymbol{\pi'}_j^{(l)}$.

- Let B be the set of parties corrupted by $\mathcal{A}$.
- Let $\mathcal{B}_1, \mathcal{B}_2$ be the algorithms associated with ideal switching played against $\mathcal{A}$.
- Compute $(\mathsf{pk}', \mathsf{sk}_1, \ldots, \mathsf{sk}_N) \leftarrow \mathcal{B}_1(\mathsf{pk})$.
- Perform all steps of the protocol on behalf of the honest parties except:
  1. On the j^{th} invocation of line 10 run $\mathcal{B}_2$ on inputs $\mathsf{pk}, (\mathsf{sk}_i)_{i \in \overline{B}}, \boldsymbol{\pi'}_j^{(l)}, \boldsymbol{\pi}_j^{(l)}$.
  2. In place of line 17 run $(n_1, \ldots, n_{|S_C|}) \leftarrow \mathcal{F}_{\mathsf{DEC}}(\mathsf{sk}_1, \ldots, \mathsf{sk}_N, \mathbf{v}_{\text{TALLY}})$.

Figure 8: Simulator for Protocol 7.

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

Figure 9: Suppose $c = 6$ and $k = 3$. Preference-order ballot right.

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

Figure 10: Preference-order ballot after elimination of candidate 3.

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

Table 1: Results for Sample IRV Counts

cation comes almost for free, and a switching protocol that allows a return to the cryptosystem from which more multiplications can be performed.

References

[ADPQ09] Ben Adida, Olivier De Marneffe, Olivier Pereira, and Jean-Jacques Quisquater. 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'09, pages 10–10, Berkeley, CA, USA, 2009. USENIX Association.

[AF07] Masayuki Abe and Serge Fehr. Perfect nizk with adaptive soundness. In Salil P. Vadhan, editor, *Theory of Cryptography: 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007. Proceedings*, pages 118–136, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[BBK+12] Josh Benaloh, Mike Byrne, Philip T. Kortum, Neal McBurnett, Olivier Pereira, Philip B. Stark, and Dan S. Wallach. Star-vote: A secure, transparent, auditable, and reliable voting system. *CoRR*, abs/1211.1904, 2012.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matt Franklin, editor, *Advances in Cryptology – CRYPTO 2004: 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004. Proceedings*, pages 41–55, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[BDO14] Carsten Baum, Ivan Damgård, and Claudio Orlandi. Publicly auditable secure multi-party computation. In *International Conference on Security and Cryptography for Networks*, pages 175–196. Springer, 2014. Also Cryptology ePrint Archive, Report 2014/075: `http://eprint.iacr.org/2014/075`.

[Bei96] Amos Beimel. *Secure Schemes for Secret Sharing and Key Distribution*. PhD thesis, Israel Institute of Technology, 1996.

[BFM88] Manuel Blum, Paul Feldman, and Silvio Micali. Non-interactive zero-knowledge and its applications. In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, STOC '88, pages 103–112, New York, NY, USA, 1988. ACM.

[BGN05] Dan Boneh, Eu-Jin Goh, and Kobbi Nissim. Evaluating 2-dnf formulas on ciphertexts. In *Proceedings of the Second International Conference on Theory of Cryptography*, TCC'05, pages 325–341, Berlin, Heidelberg, 2005. Springer-Verlag.

[BMN+09] Josh Benaloh, Tal Moran, Lee Naish, Kim Ramchen, and Vanessa Teague. Shufflesum: Coercion-resistant verifiable tallying for stv voting. *Trans. Info. For. Sec.*, 4(4):685–698, December 2009.

[BN06] P. S. L. M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography – SAC'2005*, volume 3897 of *LNCS*, pages 319–331. Springer, 2006.

[CB87] Josh Cohen Benaloh. Secret sharing homomorphisms: Keeping shares of a secret secret. In *Proceedings on Advances in cryptology—CRYPTO '86*, pages 251–260, London, UK, UK, 1987. Springer-Verlag.

[CF15] Dario Catalano and Dario Fiore. 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 '15, pages 1518–1529, New York, NY, USA, 2015. ACM.

[CGGI13] Véronique Cortier, David Galindo, Stéphane Glondu, and Malika Izabachène. Distributed elgamal à la pedersen: Application to helios. In *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, WPES '13, pages 131–142, New York, NY, USA, 2013. ACM.

[CIL17] Guilhem Castagnos, Laurent Imbert, and Fabien Laguillaumie. Encryption switching protocols revisited: Switching modulo p. Cryptology ePrint Archive, Report 2017/503, 2017. `http://eprint.iacr.org/2017/503`. To appear in CRYPTO '17.

[CM99] Jan Camenisch and Markus Michels. Separability and efficiency for generic group signature schemes. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 15–19, 1999 Proceedings*, pages 413–430, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[CPP16] Geoffroy Couteau, Thomas Peters, and David Pointcheval. Encryption switching protocols. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016: 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, pages 308–338, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[DG03] Ivan Damgrd and Jens Groth. Non-interactive and reusable non-malleable commitment schemes. Cryptology ePrint Archive, Report 2003/080, 2003. `http://eprint.iacr.org/2003/080`.

[DPSZ11] I. Damgard, V. Pastro, N.P. Smart, and S. Zakarias. Multiparty computation from somewhat homomorphic encryption. Cryptology ePrint Archive, Report 2011/535, 2011. `http://eprint.iacr.org/2011/535`.

[DSDCO+01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. *Robust Non-interactive Zero Knowledge*, pages 566–598. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for diffie-hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 129–147, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[Fel87] Paul Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proceedings of the 28th Annual Symposium on Foundations of Computer Science*, SFCS '87, pages 427–438, Washington, DC, USA, 1987. IEEE Computer Society.

[Fre10] David Mandell Freeman. 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*, EUROCRYPT'10, pages 44–61, Berlin, Heidelberg, 2010. Springer-Verlag.

[GG05] Eu-Jin Goh and Philippe Golle. Event driven private counters. In *Proceedings of the 9th International Conference on Financial Cryptography and Data Security*, FC'05, pages 313–327, Berlin, Heidelberg, 2005. Springer-Verlag.

[GJKR07] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin. Secure distributed key generation for discrete-log based cryptosystems. *J. Cryptol.*, 20(1):51–83, January 2007.

[GOS06] Jens Groth, Rafail Ostrovsky, and Amit Sahai. Non-interactive zaps and new techniques for nizk. In Cynthia Dwork, editor, *Advances in Cryptology - CRYPTO 2006: 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006. Proceedings*, pages 97–111, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[Gro10] Jens Groth. Short non-interactive zero-knowledge proofs. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010: 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, pages 341–358, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In *Proceedings of the Theory and Applications of Cryptographic Techniques 27th Annual International Conference on Advances in Cryptology*, EUROCRYPT'08, pages 415–432, Berlin, Heidelberg, 2008. Springer-Verlag.

[Hea07] James Heather. Implementing stv securely in prêt à voter. In *Proceedings of the 20th IEEE Computer Security Foundations Symposium*, CSF '07, pages 157–169, Washington, DC, USA, 2007. IEEE Computer Society.

[HHH+14] Gottfried Herold, Julia Hesse, Dennis Hofheinz, Carla Ràfols, and Andy Rupp. Polynomial spaces: A new framework for composite-to-prime-order transformations. Cryptology ePrint Archive, Report 2014/445, 2014. `http://eprint.iacr.org/2014/445`.

[Ped91] Torben Pryds Pedersen. A threshold cryptosystem without a trusted party. In *Proceedings of the 10th Annual International Conference on Theory and Application of Cryptographic Techniques*, EUROCRYPT'91, pages 522–526, Berlin, Heidelberg, 1991. Springer-Verlag.

[PIK94] Choonsik Park, Kazutomo Itoh, and Kaoru Kurosawa. Efficient anonymous channel and all/nothing election scheme. In *Workshop on the Theory and Application of Cryptographic Techniques on Advances in Cryptology*, EUROCRYPT '93, pages 248–259, Secaucus, NJ, USA, 1994. Springer-Verlag New York, Inc.

[RT09] Peter Y. A. Ryan and Vanessa Teague. Ballot permutations in prêt à voter. In *Proceedings of the 2009 Conference on Electronic Voting Technology/Workshop on Trustworthy Elections*, EVT/WOTE'09, pages 13–13, Berkeley, CA, USA, 2009. USENIX Association.

[Rya05] Peter Y. A. Ryan. A variant of the chaum voter-verifiable scheme. In *Proceedings of the 2005 Workshop on Issues in the Theory of Security*, WITS '05, pages 81–88, New York, NY, USA, 2005. ACM.

[Rya08] P. Y. A. Ryan. Prêt à voter with paillier encryption. *Math. Comput. Model.*, 48(9-10):1646–1662, November 2008.

[Sco02] Mike Scott. 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`.

[ST04] Berry Schoenmakers and Pim Tuyls. Practical two-party computation based on the conditional gate. In Pil Joong Lee, editor, *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*, pages 119–136, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[SV15] Berry Schoenmakers and Meilof Veeningen. Universally verifiable multiparty computation from homomorphic cryptosystems. In *International Conference on Applied Cryptography and Network Security*, pages 3–22. Springer, 2015. Cryptology ePrint Archive, 2015/058: `http://eprint.iacr.org/2015/058`.

[Wat11] Brent Waters. 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*, PKC'11, pages 53–70, Berlin, Heidelberg, 2011. Springer-Verlag.

A 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 $l = 1$.

$\mathsf{Setup}(1^\lambda)$: 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 10. 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 $\{g^{\vec{x}_1}, \ldots, g^{\vec{x}_l}\}$ and $\{h^{\vec{x}'_1}, \ldots, h^{\vec{x}'_l}\}$ 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 2.3.

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

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

$\mathsf{Multiply}_{\mathsf{src}}(\mathsf{pk}, C, C')$: 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.

$\mathsf{Add}_{\mathsf{src}}(\mathsf{pk}, C, C')$: 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 $C'' = (C''_0, C''_1)$.

$\mathsf{Add}_{\mathsf{tgt}}(\mathsf{pk}, C, C')$: 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''.

$\mathsf{Dec}_{\mathsf{src}}(\mathsf{sk}, C)$: 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.

$\mathsf{Dec}_{\mathsf{tgt}}(\mathsf{sk}, C)$: 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 2 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 $(u_i)_{1 \le i \le l+1}$ and $(b_i)_{1 \le i \le l+1}$ 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 $(u_i)_{1 \le i \le l+1}$ and $(u'_i)_{1 \le i \le l+1}$ 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 9. *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 $(\mathbb{G}, g, g^A, g^{\vec{v}})$, attacker $\mathcal{B}$ performs the following steps. Write $g^A = g^{\vec{x}_1} \| \ldots \| g^{\vec{x}_l}$. Set $G_1 = \{g^{\vec{x}_1}, \ldots, g^{\vec{x}_l}\}$. 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 $C_0 = \mathbf{g}^{M_\beta} \cdot g^{\vec{v}}, C_1 = \mathbf{h}^{M_\beta} \cdot \mathbf{h}_1 : \mathbf{h}_1 \in_r H_1$. Send CT to $\mathcal{A}$. Output the bit that $\mathcal{A}$ outputs. Clearly $\Pr[1 \leftarrow \mathcal{B}(\mathbb{G}_1, g, (g^{\vec{x}_1}, \ldots, g^{\vec{x}_l}), g^{\sum_{j=1}^{l} a_j \vec{x}_j})] = \Pr[1 \leftarrow \mathcal{A}(H_0)]$, while $\Pr[1 \leftarrow \mathcal{B}(\mathbb{G}_1, g, (g^{\vec{x}_1}, \ldots, g^{\vec{x}_l}), g^{(u_1, \ldots, u_{l+1})^T})] = \Pr[1 \leftarrow \mathcal{A}(H_1)]$. Thus $Adv_{\mathcal{B}} = |\Pr[1 \leftarrow \mathcal{A}(H_0)] - \Pr[1 \leftarrow \mathcal{A}(H_1)]| = Adv_{\mathcal{A}} = \epsilon_1$. □

Proposition 10. *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 $(\mathbb{G}_2, h, h^A, h^{\vec{v}'})$, attacker $\mathcal{B}$ performs the following steps. Write $h^A = h^{\vec{x}'_1} \| \ldots \| h^{\vec{x}'_l}$. Set $H_1 = \{h^{\vec{x}'_1}, \ldots, h^{\vec{x}'_l}\}$. 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 $C_0 = \mathbf{g}^{M_\beta} \cdot (g^{u_1}, \ldots, g^{u_{l+1}}) : u_1, \ldots, u_{l+1} \in_R \mathbb{Z}_p, C_1 = \mathbf{h}^{M_\beta} \cdot h^{\vec{v}'}$. Send CT to $\mathcal{A}$. Output the bit that $\mathcal{A}$ outputs. Clearly $\Pr[1 \leftarrow \mathcal{B}(\mathbb{G}_2, h, (h^{\vec{x}'_1}, \ldots, h^{\vec{x}'_l}), h^{\sum_{j=1}^{l} b_j \vec{x}'_j})] = \Pr[1 \leftarrow \mathcal{A}(H_1)]$, while $\Pr[1 \leftarrow \mathcal{B}(\mathbb{G}_2, h, (h^{\vec{x}'_1}, \ldots, h^{\vec{x}'_l}), h^{(u'_1, \ldots, u'_{l+1})^T})] = \Pr[1 \leftarrow \mathcal{A}(H_2)]$. Thus $Adv_{\mathcal{B}} = |\Pr[1 \leftarrow \mathcal{A}(H_1)] - \Pr[1 \leftarrow \mathcal{A}(H_2)]| = Adv_{\mathcal{A}} = \epsilon_2$. □

Proposition 11. *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. □

B NIZKs

In this section we present non-interactive zero knowledge proofs for the relations described in Section 5. For convenience our presentation of these is unified - our proofs assume inputs under the verifiable commitment scheme of [GOS06] while known techniques can be used to interchange between commitments under this scheme and ciphertexts under the other cryptosystems in this paper [CM99].

Definition 15 (Decision Linear Assumption [BBS04])**.** *Let g, u, v, h be generators in $\mathbb{G}$. For adversary $\mathcal{A}$ define*

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

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

Decision Linear Commitments [GOS06] We present the homomorphic commitment scheme of [GOS06] 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 $\mathbb{G}$ for which a symmetric bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ exists. The reason for this modification will become apparent when present the NIZK proof systems themselves.

Setup :

Let $(p, \mathbb{G}, e, g) \leftarrow \mathcal{G}(1^\lambda)$. Let $\mathbf{g} \in_R \mathbb{G}^2$. Let $x, y \to \mathbb{Z}_p^*$. 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. $\mathbf{u}, \mathbf{v} \in_R \mathbb{G}^2$. Let $\mathbf{w} = \mathbf{u}^{1/x}\mathbf{v}^{1/y}$.
2. Return $ck = (\mathsf{pk}, \mathbf{u}, \mathbf{v}, \mathbf{w})$.

Perfectly binding key generation K_{bind} :

1. $\mathbf{u}, \mathbf{v}, \mathbf{w} \in_R \mathbb{G}^2$.

2. Return $ck = (\mathsf{pk}, \mathbf{u}, \mathbf{v}, \mathbf{w})$.

Commitment :

To commit to message $m \in \mathbb{Z}_p$ do

1. $r, s \leftarrow \mathbb{Z}_p$
2. Return $c = (c_1, c_2, c_3) = \mathsf{com}(m; r, s) = (\mathbf{u}^m \mathbf{f}^r, \mathbf{v}^m \mathbf{h}^s, \mathbf{w}^m \mathbf{g}^{r+s})$.

Trapdoor opening :

Given a commitment $c = \mathsf{com}(m; r, s)$ under a perfectly hiding key, we have $c = \mathsf{com}(m'; r - (m' - m)x, s - (m' - m)y)$. Thus we can create a perfectly hiding commitment and open it to any value we wish if we have the trapdoor key (x, y).

B.1 Plaintext Equivalence Proof [AF07]

Proof

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 m under pk_1 and pk_2.

Prover's Input : (m, r, s, r', s') so that $c_1 = (\mathbf{f}^r \mathbf{u}_1^m, \mathbf{h}^s \mathbf{v}_1^m, \mathbf{g}^{r+s} \mathbf{w}_1^m)$, $c_2 = (\mathbf{f}^{r'} \mathbf{u}_2^m, \mathbf{h}^{s'} \mathbf{v}_2^m, \mathbf{g}^{r'+s'} \mathbf{w}_2^m)$.

Proof :

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

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

Verifier : Check that

$$\begin{aligned}
e(\mathbf{f}, \pi_1) &= e(\mathbf{g}, c_{11} c_{22}^{-1} \pi_2) \\
e(\mathbf{h}, \pi_1 \pi_3 c_{13} c_{23}^{-1} c_3) &= e(\mathbf{g}, c_{11} c_{22}^{-1} \pi_4)
\end{aligned}$$

B.2 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

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 $[m]_j$ be the j^{th} bit of integer m.

Statement : c is a commitment to m under $(\mathbf{u}, \mathbf{v}, \mathbf{w})$ and $m \in [0, 2^\lambda - 1]$.

Prover's Input : m, r, s so that $c = (\mathbf{f}^r \mathbf{u}^m, \mathbf{h}^s \mathbf{v}^m, \mathbf{g}^{r+s} \mathbf{w}^m)$.

Proof : For $0 \leq j < \lambda$ let

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

Let $\pi' = \mathbf{g}^{r-\sum_{j=0}^{\lambda-1} 2^j \cdot r_j}$.

Send $(c_j, \pi_{1j}, \pi_{2j}, \pi_{3j})_{0 \leq j < \lambda}$, 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 12. *The above range proof is perfectly complete, perfectly sound and is computational zero knowledge if the Decision Linear assumption holds. The proof consists of* $2 + 12\lceil \log_2 m \rceil$ *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 $\mathbf{u} \in \mathbb{Z}_p, \mathbf{v} \in \mathbb{Z}_p, \mathbf{w} = \mathbf{u}^{1/x}\mathbf{v}^{1/y}$. The simulator sets $\mathbf{f} = \mathbf{g}^x, \mathbf{h} = \mathbf{g}^y : x, y \in_r \mathbb{Z}_p$ 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 $r_j, s_j \in \mathbb{Z}_p$ 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 $\pi' = (c_1 \cdot (\prod_{j=0}^{\lambda-1}(c_{1j})^{2^j})^{-1})^{1/x}$. 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 $c_j = \mathsf{com}(m_j; r_j, s_j)$ for some m_j. By the perfect binding property of the commitment scheme, these are all unique. Then valid π_{1j} implies that $e(\mathbf{f}, \pi_{1j}) = e(\mathbf{u}, \mathbf{u})^{m_j(m_j-1)} e(\mathbf{f}, \mathbf{u}^{r_j(2m-1)}) e(\mathbf{f}, \mathbf{f}^{r_j^2})$. If $\mathbf{u} \notin \text{span}(\mathbf{f})$, it follows that $m_j(m_j - 1) = 0$, thus $m_j = 0$ or $m_j = 1$. 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 $m_j = 0 \vee m_j = 1$ 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 $c_3 = w^m g^{r'+s'} : m = \sum_{j=0}^{\lambda-1} m_j 2^j$. Hence $m \in [0, 2^\lambda - 1]$.

Protocol π_{COND} ([ST04])

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 $\varepsilon_i \leftarrow \mathcal{P}_{\mathsf{mul}}(\langle\langle s_i \rangle\rangle, [x_{i-1}], [x_i]), \mathcal{P}_{\mathsf{mul}}(\langle\langle s_i \rangle\rangle, [y_{i-1}], [y_i])$. 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.

Figure 11: Protocol π_{COND} computing the conditional gate.

C Distributed Key Generation Sub-Protocols

C.1 Conditional Gate

Schoenmakers et al. [ST04] 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 [ST04] 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 12. This sequence forms the basis for our protocol in Section C.3 which computes the private xor-sum of a number of encrypted inputs.

Theorem 1 [ST04] 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 15.

C.2 Private Sum Modulo Two

Theorem 13. *Protocol π_{SUM} securely computes encryption switching in the $\mathcal{F}_{\mathsf{COND}}$-hybrid model against statically chosen adversaries if $\mathcal{P}_{\mathsf{bit}}$ is a secure NIZK proof system.*

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 $\varepsilon_{ij} \leftarrow \mathcal{P}_{\mathsf{bit}}(c_{ij})$.
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 $\vec{C} = (C_0, \ldots, C_{\lambda-1})$.

Figure 12: Protocol π_{SUM}.

Simulator for π_{SUM}

Input : $([u_1], \ldots, [u_N], \vec{C} = (\tilde{C}_0, \ldots, \tilde{C}_{\lambda-1})$

- 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 $\mathcal{F}_{\mathsf{COND}}(c'_{Nj}, C_j)$.

Figure 13: Simulator for protocol π_{SUM}.

Protocol π_{DEC}
Common Input : C under Enc_1.
Private Input : P_i holds a share of secret key, sk_i

Player P_i :

1. Publishes $C_i \leftarrow C^{\mathsf{sk}_i}$.
2. Outputs $m \leftarrow \prod_{i=1}^{N} C_i$.

Figure 14: Protocol π_{DEC}.

Simulator for π_{DEC}
Input : C, m

- Let P_N be honest.

1. Perform the above steps for honest parties except P_N.
2. P_N publishes $C_N \leftarrow \mu / \prod_{i=1}^{N-1} C_i$.

Figure 15: Simulator for protocol π_{DEC}.

Proof. Let $(c^*_{ij}, \varepsilon^*_{ij}, C^*_j, {c'_{ij}}^*, {c''_{ij}}^*)_{1 \le i \le N}$ 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. □

C.3 Decryption Protocol

Lemma 14. *Protocol* π_{DEC} *unconditionally computes encryption switching against statically chosen adversaries.*

Proof. Since we are in the static adversarial model we may assume without loss of generality some player, e.g., player N is honest. Then P_N's output, C_N, in the simulation satisfies $\mu = (\prod_{i=1}^{N-1} C_i) \cdot C_N = \prod_{i=1}^{N} C_i$ which is identically distributed to that in the real protocol. The

proof follows. □

D Parameter Selection for Distributed Key Generation

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

Lemma 15. *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 $\gamma_k = \sum_{i,j\in Q}\sum_{f+g=k}\alpha_{if}\alpha'_{jg} + \sum_{i\in Q}\beta_{ik} = \sum_{i,j\in Q} a_{ij} + \sum_{i\in Q} b_i = \sum_{i,j\in Q\setminus C} a_{ij} + \sum_{(i,j)\in C\times Q\cup Q\times C} a_{ij} + \sum_{i\in Q\setminus C} b_i + \sum_{i\in C} b_i = X + \overline{X} + Y + \overline{Y}$. Clearly $|X| \le (|Q|-T)^2\cdot\ell\cdot 2^{2\lambda_A}$. On the 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 $SD(X, X+Y) \ge 1-2^{-\lambda}$ holds when

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

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

E Proving Correct Decryption

Protocol π_{SWITCH} uses a proof of correct decryption, which follows from the solution proposed in [SV15] for instance.

Protocol π_{CD} **([SV15])**
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 : $\Sigma.\mathsf{ver}(d, d_i, v, v_i; a, b; c; r) := d^r \stackrel{?}{=} a(d_i)^c \wedge v^r \stackrel{?}{=} b(v_i)^c$
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** $(d^r(d_i)^{-c}, v^r(v_i)^{-c}; c; r)$

F Definition of Universally Verifiable secure computation

The following ideal model for universally verifiable encryption switching is derived from [SV15], Process 5. A protocol is defined to implement verifiable secure function evaluation if, for every

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 ▷ Honest inputs.

$\{sk_i\}_{i \in B} :=$ recv$(\mathcal{S})$ ▷ Corrupted inputs.

if $|B| \geq t$ **then**

  Send r_i to $\mathcal{S}$ for all i. ▷ Threshold corrupt. ▷ Computation phase.

  $E = \mathsf{Enc}_{tgt}(m)$

end if

if $\mathcal{R} \notin B$ **then** ▷ 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 ▷ 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 ▷ Proof phase.

  if $\mathcal{V} \notin C$ then send$(R, \mathcal{V})$.

end if

Figure 16: Ideal party for verifiable π_{SWITCH}.

Ours	$(8c(k^2-k)M + 6kcN) \times op_{ExpDH} + (128N^2 + 32N) \times op_{ExpCF}$
[DPSZ11]	$7c(k^2-k)M(N \times op_{AAdd} + 2 \times op_{AMult})$

Table 2: Cost comparison to [DPSZ11].

probabilistic polytime real-world adversary $\mathcal{A}$, there exists a probabilistic polytime ideal-world adversary $\mathcal{S}_{\mathcal{A}}$ s.t., for all inputs I; all sets of corrupted parties B and all auxiliary input a, the adversary's view of the real execution is computationally indistinguishable from the simulated ideal execution. The ideal model assumes a receiving party $\mathcal{R}$, who may be every member of the public.

G Cost Comparison to [DPSZ11]

Table 2 compares the total arithmetic operations used in our scheme to [DPSZ11]. 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 $\ell = 8$.