Share & Shrink: Ad-Hoc Threshold FHE with Short Ciphertexts and its Application to Almost-Asynchronous MPC

Antoine Urban and Matthieu Rambaud

Telecom Paris, Institut Polytechnique de Paris, France

Abstract. We consider protocols for secure multi-party computation (MPC) under honest majority, i.e., for $N = 2t + 1$ players of which t are corrupt, that achieve *guaranteed output delivery* (GOD), and in *constant latency*, independently from the circuit and N. A generic approach to this problem requires at least 3 consecutive broadcasts in the plain model without PKI. State-of-the-art protocols with 2 consecutive broadcasts, namely [GLS, Crypto'15] and [BJMS, Asiacrypt'20], however, suffer from a large size of threshold homomorphic ciphertexts. We aim for more efficient protocols in 2 broadcasts, that subsequently enjoy a *Responsive execution*, i.e., at the speed of the network.

To achieve this goal, we design a new approach with short threshold fully homomorphic (FHE) ciphertexts, which in turn impacts the computational complexity. The main building block of our technique is a threshold encryption scheme which is Ad-Hoc, i.e., which only takes as parameter N public keys independently generated, equipped with a threshold shrinking mechanism into threshold FHE ciphertexts.

One ingredient of independent interest is a linear secret sharing over RLWE rings with arbitrary modulus. By contrast, previous threshold FHE required the modulus to be prime and at least as large as $N + 1$.

Another significant advantage of this approach is that it also allows an arbitrary number of lightweight *external input owners* to feed their inputs in the computation by simply encrypting them with the Ad-Hoc scheme, then go offline.

We finally prove the impossibility of 1-Broadcast-then-Asynchronous MPC for $N \leq 3t - 4$, showing tightness of our 2 broadcasts.

1 Introduction

Secure multi-party computation (MPC) protocols aim to achieve the most general goal in cryptography: implementing an ideal functionality that collects private inputs from N players, computes an evaluation of any given arithmetic circuit on these inputs, and returns it to the players. This functionality should behave unchanged even if up to t players, denoted as *maliciously corrupt*, are completely controlled by a computationally bounded adversary $\mathcal{A}$. One issue in making it practical is achieving **Constant Latency to Output**, on which we focus in this paper. Protocols with this property must guarantee that the

number of consecutive interactions to output is independent of N and of the circuit to be computed. The main MPC technique for achieving constant latency with short communication, independently of the width of the circuit, relies on Fully Homomorphic Encryption (FHE). On the one hand, the technique denoted multi-key (MFHE) [CDKS19; MTBH20], enables joint evaluation of encryptions under different keys. On the other hand, any aborting player prevents subsequent decryption, which prevents the following desirable guarantee to hold in constant latency (nor Responsively): **Guaranteed Output Delivery (GOD), under honest majority $N = 2t + 1$**, i.e., that every honest player should output with overwhelming probability in any execution, regardless of the adversary.

The main approach to enabling GOD is to use what is referred to as a (N, t)-Threshold Encryption scheme. Following a definition which may be seen as "canonical", e.g., as in [Bon+18, §6.2], it consists of: a setup, i.e., the generation of a common public encryption key, denoted *Threshold Key*, along with the private assignment to each player of a secret *Decryption Key Share* such that ciphertexts have IND-CPA for any adversary controlling up to t key shares; an algorithm that takes as input a key share and any ciphertext, and outputs a *Partial Decryption*; and finally an algorithm that takes any $t + 1$ correct partial decryptions of any ciphertext and Combines them into the plaintext. The task of establishing a threshold key and assigning key shares must be, accordingly, securely implemented under honest majority. Protocols for this task are known as "Distributed key Generation" (DKG). There are some shortcomings in actual Threshold FHE (TFHE), however. Sometimes the DKG is not addressed, as in [Ash+12] & [Bon+18, §5], or, is established by a single entity, as in [Bon+18, §6.2], thereby preventing security under honest majority. Even when a threshold key is set-up, aborting players can delay [Ash+12] or completely prevent output delivery [Kim+20; Par21] in some protocols: see Table 1, §1.3 and §A. Finally, the choices of parameters in TFHE appear more constrained than in their single-key counterparts, e.g., imposing the modulus q of the ciphertext space to be a prime and at least as large as $N + 1$ [Ash+12][GLS15][Bon+18, §B][Kim+20].

"Almost Asynchronous": Small Latency to Responsiveness. Responsiveness refers to the ability of a protocol to run at the actual speed of the network [CGHZ16; PR18], see more in §B.1. In our context, we say that a (part of) protocol, or a (part of) an execution, is *Responsive* if: communications consist only in sending messages over point-to-point secure channels with eventual output delivery, and, every player sends its next message every time it receives a set of messages from $N - t$ senders that all "look honest". However, it is trivial that protocols allowing players to output after an execution responsive since the beginning, do not withstand more than $t < N/3$ whatever the setup, e.g., see [BHN10, p2]. Therefore, to enable honest majority, preliminary non-responsive events must be appended at the beginning of the execution. They typically consist in synchronous communications, if not broadcasts. We refer to the number of consecutive pre-responsive events as the *latency to responsiveness*. The protocol denoted "almost asynchronous" in [BHN10] requires only one non-responsive preliminary event, which is that all honest players broadcast their inputs en-

crypted under a single threshold key. However their protocol lacks a DKG and has non-constant latency. One the one hand, achieving constant latency could be made be specifying their threshold encryption to be a TFHE. On the other hand, ignoring the additional technical complications of TFHE, the task of DKG is best known for taking 2 consecutive events without a PKI, since [FS01]: players publish public keys, then broadcast publicly verifiable secret sharings of their contributions to the secret threshold key. Overall, we are left with a Generic Approach for MPC which would take a total of 3 consecutive non-responsive events. Thus it is a question to get a smaller latency than 3 to responsiveness. See also [FN09; PR18] for related contexts.

Short Ciphertexts. In a breakthrough approach, [GLS15], the latency to responsiveness is reduced to 2, pre-pended with reception of a uniform string. Then, the very original approach of [BJMS20] furthermore enables provision of external inputs (see below). The problem in [GLS15; BJMS20] is that the sizes of their TFHE ciphertexts, used for evaluation, are very large. This issue impacts the works using their schemes [GPS19; Dam+21]. Our main question is therefore:

Can we allow latency 2 *to Responsiveness, while keeping small the sizes of homomorphically evaluated ciphertexts ?*

Provision of External Inputs. [Bar+18] point out that the lack of handling inputs from external lightweight owners, like mobile phones or web browsers, is one of the main obstacles to the deployment of MPC in practice. Thus, we consider the following strictly more general model. Inputs come from arbitrarily many entities, possibly all corrupt, denoted $\mathscr{L}$ as (Lightweigth) "Input Owners", which are logically distinct machines from players, although some may physically run on the same hardware. We define *Provision of External Inputs* as the following requirements: (i) Owners need only broadcasting one (encrypted) message to the players, then can go offline; (ii) The material needed to encrypt boils down to keys and strings noninteractively generated and published by players (§3, §5.6); (iii) Without the need of verifying this material. This is however not enabled by [GLS15] nor by the Generic approach, whereas in [BJMS20] the size of ciphertexts depend quadratically on the number of input owners. We thus ask

Can we furthermore allow provision of external inputs?

1.1 Results

We answer positively to the above questions by proposing a new approach. We first state the results, then explain the technical contributions in §1.2. We denote by R_q the ciphertext space of the RLWE-based FHE known as BFV [FV12], its elements are encoded in size $O(n \log q)$ bits, where n is the dimension of the ring and q the modulus, see §4 for details. We assume an external functionality $\overline{\mathcal{G}}_{\mathrm{URS}}$ that samples a uniform string of bits of pre-defined length, which can be

safely re-used in other concurrent executions. Uniformity of the sampling allows much more efficient implementations, e.g., with MPC under honest majority, or, more heuristic "nothing up my sleeve" public coin tossing, than if the string had been specified to have a hidden structure. We assume a Terminating Reliable Broadcast functionality $\mathcal{F}_{\mathrm{BC}}$, which guarantees eventual output whatever the (non)behavior of the sender. The first kind of instance of $\mathcal{F}_{\mathrm{BC}}$ is denoted "bulletin board PKI" bPKI, and is used by players to broadcast their public keys towards both players and input owners. The second one, denoted BC, is used by both players and input owners, but delivers output to players only. Subsequent Responsive steps are performed only over pairwise public authenticated channels with eventual delivery.

Theorem 1 (§5.4). *Consider $N = 2t + 1$ players, of which t are maliciously corrupt by a polynomial adversary. There exists a protocol that UC implements secure evaluation of any arithmetic circuit, with GOD, latency 2 to Responsiveness, broadcast size of $O(Nn \log q)$ bits for each player and owner, followed by 2 responsive events to output, comprising homomorphic evaluation of the circuit on ciphertexts of size $O(n \log q)$. It has Provision of External Inputs.*

It has furthermore the *delayed function property*, i.e., messages from owners are independent of the circuit to be evaluated. In the model of a bare PKI before the execution starts [Dam+21], then latency to Responsiveness would be brought back to 1. In short, our protocol may provide a practical tradeoff between computation and communication as follows. On the one hand we add one Responsive step. This price becomes further marginal in the use-cases where *interactive bootstrapping* becomes advantageous. Namely, [Cho+13] introduced a mechanism that bootstraps any TFHE ciphertext at no computation cost, in one single interaction, which we observe can be performed responsively. When used in a certain regime, as evidenced in Table VII of [MTBH20], then it enables to divide the overall evaluation time.

On the other hand, we reap the benefit that the sizes of the TFHE ciphertexts, on which evaluation is performed, are orders of magnitude smaller than in previous approaches [GLS15; BJMS20], as shown in Table 1, due to a combination of at least two factors. First, the TFHE ciphertexts which we obtain are standard BFV ciphertexts encrypted under a single public (threshold) key. But, BFV has some inherent advantages over the GSW FHE [GSW13], which was used in previous approaches (i) its ciphertext sizes are one order of magnitude smaller (ii) the plaintexts are large modular integers, instead of bits, enabling efficient numerical computations (iii) several plaintexts can be encoded in one single ciphertext and evaluated in parallel. See §A.2 for more context. Second, as made clear below, our approach essentially removes other sources of extra size overhead in previous approaches.

By-Product: the First RLWE FHE Born and Raized Distributively. *Ignoring* the main construction outlined below, we are still left with a complete toolbox §5.2 for the Generic approach to TFHE-based MPC, which further-

more removes previous sources of non-GOD or inefficiencies. Namely, we provide a threshold DKG for BFV, along with threshold generation of keys for relinearization: §4.3, and bootstrapping: §5.5. Compared to previous works [Kim+20; Par21], cf. §1.3 & §A, the last two generations operate *in parallel* of the DKG, and the whole needs not be restarted in presence of deviating players.

We allow for an arbitrary ciphertext modulus q, nonwithstanding q is odd in our particular choice of parameters. Previous TFHE required q to be a prime at least as large as $N+1$. By contrast we allow in general, e.g., $q=2$, or a prime power, which can enable computational optimizations [CH18].

Protocol	Pre-Responsive events			Responsive events	GOD	Size of FHE ciphertexts	Prov. Ext. Inputs
	1^{st} Event	2^{nd} Event	3^{nd} Event				
[CDKS19]	$\overline{\mathcal{G}}_{\text{URS}}$	bPKI	BC	P2P	×	$O(N.n\log q)$	×
[Kim+20]	$\overline{\mathcal{G}}_{\text{URS}}$	bPKI + P2P	BC	P2P	×	$O(nN\log q)$	×
[MTBH20]	$\overline{\mathcal{G}}_{\text{URS}}$	bPKI + BC	BC	P2P	×	$O(n\log q)$	×
[Par21]	$\overline{\mathcal{G}}_{\text{URS}}$	bPKI	BC	P2P	×	$O(n\log q)$	×
[GLS15]	$\overline{\mathcal{G}}_{\text{URS}}$	bPKI	BC	P2P	✓	$O(n(N)^2\log^3 q)$	×
[BJMS20]	bPKI	BC	–	P2P	✓	$O((\lvert\mathscr{L}\rvert n)^2\log^3 q)$	✓
By-Product	$\overline{\mathcal{G}}_{\text{URS}}$ + bPKI	BC	BC	P2P	✓	$O(n\log q)$	×
Thm 1	$\overline{\mathcal{G}}_{\text{URS}}$ + bPKI	BC	–	2 P2P	✓	$O(n\log q)$	✓

Table 1: MPC for $N=2t+1$ players and $|\mathscr{L}|$ input owners, using FHE with lattice dimension n and modulus q. GOD stated within a constant worst-case latency, thus the non-counting of [CDKS19; Kim+20; MTBH20] (see §A.1.2). "FHE ciphertexts" are those homomorphically evaluated, so after Expansion/Transformation. $\overline{\mathcal{G}}_{\text{URS}}$, bPKI and BC denote three external functionalities introduced in 1.1. "+" for events in parallel. More details in §1.3 then §A.

Impossibility of* 1*-Broadcast-then-Asynchronous MPC. In Thm. 13 we show tightness of our latency to Responsiveness by showing that for $t\geq 3$ and $N\leq 3t-4$ then some functionalities are not securely implementable, without setup, in one synchronous round with access to a broadcast, followed by an arbitrary number of pairwise asynchronous communications. It thus parallels [PR18], which dealt with perfect security. The strategy adapts [GIKR02, §4.1].

1.2 Technical Contributions

Our method structurally differs from previous approaches, thanks to new tools of independent interest. We provide a (N,t) threshold encryption scheme, requiring for its operation the sole knowledge of N public keys which were generated and published by players independently. We thus qualify it as Ad-Hoc, following [RSY21]. Owners encrypt-then-broadcast their inputs, then go offline. Our main tool is that we equip the Ad-Hoc Threshold Encryption with a Threshold mechanism enabling players to Transform, in one Responsive step, Ad-Hoc ciphertexts

into BFV ciphertexts encrypted under any single key, thereby Shrinking their sizes and enabling homomorphic evaluation. The "any" is important, since when owners encrypt their inputs they have no knowledge of what will be the threshold key, since it is being generated by players in parallel.

The claimed Transformation is enabled by the observation that the deterministic part of BFV encryption is a *linear function* over the ring of ciphertexts, i.e., public coefficients are given by the public key, while secret variables are the secret plaintext, randomness and noise. Thus, we instantiate the Ad Hoc Encryption from a publicly Verifiable Secret Sharing (PVSS), cf [GMW91, §3.2], furthermore linear over RLWE rings.

What makes the approach structurally efficient is that, by perfect privacy of secret sharing, the description and analysis can be carried-out in the lens of single-key BFV, modulo adversarial influence on the public threshold key. To make this reduction clearer, we perform it via an intermediary gadget ideal functionality, denoted $\mathcal{F}_{\mathrm{DLC}}$, that captures the aforementioned ability to compute a delayed linear combination in one step on several PVSS. This gadget also applies to efficient distributed decryption, detailed in §5.2.2 which is more involved.

Finally, let us comment on the two reasons enabling an arbitrary modulus, which are techniques of independent interest. First, our construction of linear secret sharing over Galois extensions of RLWE rings, done in §3.1, which generalizes the case of $\mathbb{Z}/p^e\mathbb{Z}$ studied in [Abs+19]. Second, a distributed decryption technique §5.2.2, which structurally removes the need for a prime q in other approaches, as discussed in [Bon+18, §B]. We imported this generation from [GLS15], over our linear secret sharing. This generation also yields a $N!$ times smaller noise in the decrypted ciphertext, than in previous approaches.

1.3 Related Works

Further details of related works can be found in §A.

In the Approach of [GLS15] each player P_i encrypts its input under a public key encryption scheme, specific to i, which may be denoted $\mathrm{Flexible}_i$. Flexible ciphertexts are subsequently transformed into THFE ciphertexts. Since each P_i knows the secret key of $\mathrm{Flexible}_i$, none of them is threshold, preventing their safe use by external owners. Since $\mathrm{Flexible}_i$ encryption consists in a vector of GSW encryptions, some under adversarial keys, which all re-use the same secret randomness, the sizes of public keys and randomness are thus scaled larger, in order to apply the leftover-hash-lemma (LHL). However this technique is not efficiently transposable to our RLWE setting, see §A.1.1 for more details. The $O(n(N))$ appearing in Table 1 is from their §4.2.

The Ingenious Framework of [BJMS20] can be safely extended to external inputs, as follows. Each Owner encrypts its input under a locally generated key compatible with MFHE, then provides to the N players a cleverly-crafted secret sharing of the secret key, due to [Bon+18, §6], then goes offline. Then players evaluate a circuit over the $|\mathscr{L}|$ MFHE ciphertexts. The miracle is then that all the $|\mathscr{L}|$ secret sharings legated by Owners, enable players to *emulate* MFHE-reconstruction, as if it would have been performed by the $|\mathscr{L}|$ owners

themselves, in a simulatable way. The drawback of this approach is that MFHE ciphertexts unavoidably undergo a processing expanding their size in $|\mathscr{L}|$, even in $|\mathscr{L}|^2$ in their construction that uses GSW.

Generic approach. [Kim+20] carried-out the aforementioned Generic approach, with the BFV HE [FV12]. However, their distributed generation of a relinearization key, in III, terminates only if all players which contributed to the threshold encryption key, also participate in this process, thus preventing GOD. The same issue appears in the (N, N)-TFHE of [Par21]. More generally, an inherent limitation of the Generic approach is that the encryption key produced by a DKG is not published explicitly: it is the result of a local computation made by each player, which furthermore involves checking NIZKs of correctness of broadcasts in the DKG. This thus prevents Provision of external inputs. It is not hard to imagine a fix to this problem, but at the cost of one extra non-Responsive step, which would thus make a total of 4.

Other approaches. [Ana+18, §6.1] achieve MPC with GOD and delayed function property in 3 rounds. Contrary to TFHE-based approaches, the per-player communication cost depends on $|C|$ the size of the circuit evaluated, since it is in $O(N^\tau|C| + N^{\tau+1}d)$, where $\tau > 2$ and d is the depth of C. Provision of external inputs is prevented by the fact that messages contain hardcoded information combining both material specific to the player P sending them, and its secret input. We discuss further related works using Garbled circuits in §A.2, and how they benefit from being combined with TFHE, e.g., in use-cases of deep neural networks. The protocols [DHL21] [GPS19] proceed by intervals of fixed duration, denoted rounds, they could equivalently be casted as consecutive instances of bPKI [GMPS21]. The protocol [Liu+20] has Responsiveness and GOD provided up to $t_a < N/3$ corruptions, assuming a TFHE setup. It also offers trade-offs under synchrony, but at the cost of further lowering this t_a threshold. See §A.3 for related protocols having non-constant latency.

2 Model

More details and comments can be found in §B.

General notations. All logarithms are in base two, excepted in §3.1. We denote $\langle \mathbf{u}, \mathbf{v} \rangle$ the usual dot product of two vectors $\mathbf{u}, \mathbf{v}$. For two vectors $\mathbf{u}, \mathbf{v}$ we denote $\langle \mathbf{u}, \mathbf{v} \rangle$ the dot product and, for a third vector $\mathbf{w}$, we denote $\mathbf{u}^{\langle \cdot \rangle}(\mathbf{v}, \mathbf{w}) := (\langle \mathbf{u}, \mathbf{v} \rangle, \langle \mathbf{u}, \mathbf{w} \rangle)$. We denote $x \leftarrow \mathscr{D}$ the sampling of x according to distribution $\mathscr{D}$. Cardinality of a set X is denoted as $|X|$. For a finite set E, we denote $U(E)$ the uniform distribution on E. The set of positive integers $[1, \ldots, N]$ is denoted $[N]$. We denote by λ the security parameter throughout the paper. $\{0,1\}^*$ denotes bitstrings of arbitrary lengths.

We consider a positive integer n, denoted as *lattice dimension*; a monic polynomial f of degree n; $k < q$ positive integers denoted plaintext and ciphertext moduli; and $R := \mathbb{Z}[X]/f(X)$. They will be further specified in §4. We denote $R_k = R/(k.R)$ and $R_q = R/(q.R)$ the residue rings of R modulo k and q. Unless otherwise specified, we consider arithmetics in R_q. All linear forms are over R_q,

they are succinctly specified as *formal linear combinations*, e.g., let $(\overline{x_i})_i$ denote *labels* of some variables $(x_i)_i$, then, $\sum_i \lambda_i \overline{x_i}$ denotes $\left\{(x_i)_i \to \sum_i \lambda_i x_i\right\}$.

2.1 Players, Input Owners and Corruptions. We consider $N = 2t + 1$ players $\mathcal{P} = (P_i)_{i\in[N]}$, which are probabilistic polynomial-time (PPT) machines, of public identities. We also consider PPT machines denoted *input owners* $(Q_\ell)_{\ell\in\mathscr{L}}$, which are logically disjunct from players. We consider the Universal Composability (UC) model [Can01] with static corruptions, recalled in §B.8. We consider a PPT machine, denoted as the Environment $\mathscr{E}$. It fully controls an entity denoted the "dummy adversary" $\mathcal{A}$. In turn, at the beginning of the execution, $\mathcal{A}$ may statically corrupt up to t players of its choice, along with an arbitrary number of owners. In particular, our model needs not counting in the corruption budget t the honest players whose hardware is hosting a corrupt owner. Without loss of generality we assume that $\mathcal{A}$ corrupts t players, whose we denote the indices by $\mathcal{I} \subset [N]$. The remaining ones are denoted "honest" and indexed by $\mathcal{H} = [N] \setminus \mathcal{I}$. $\mathcal{A}$ notifies $\mathscr{E}$ of every message received by corrupt players and from (simulated) functionalities. We consider semi-malicious corruptions, further specified in §2.3, then compile our results into malicious security in §5.6.

2.2 Ideal Functionalities with Public Delayed Output and Formalism of GOD. Further formalism and discussions can be found in §B.

The following ideal functionalities have delivery delays of each of their outputs which are adaptively scheduled by the adversary. However, fitting this condition, known as "eventual delivery", in the UC framework, in which clocks do not exist, requires much bootstrapping [Liu+20; CGHZ16]. Thus, in order not to obfuscate our contributions, we take the following simpler approach. First, we slightly extend the formalism denoted Public Delayed Output, from [Can01]. Then, we define the ideal functionalities in this formalism. Then, we define MPC as having GOD in this formalism. Finally, we argue why our definition of GOD coincides with the classical definition, as soon as our broadcast and bulletin board are instantiated with terminating broadcast protocols, and our authenticated channels instantiated with eventual delivery.

We say that an ideal functionality $\mathcal{F}$ sends a *public delayed output* v to R if it, first, asks $\mathcal{A}$ for permission to output v to R. The adjective "public" denotes that the value v is given to the adversary. Since all our functionalities have public output, we leave it implicit in what follows. When $\mathcal{A}$ allows, then $\mathcal{F}$ outputs v to R. In addition, functionalities may request $\mathcal{A}$ to for the value of v, which we formalize as $\mathsf{network}$. Then $\mathcal{A}$ may provide it, which we formalize as $(\mathsf{activate}, v)$

The session identifier of the MPC protocol, sid, is left implicit in all calls to functionalities. Some calls to functionalities are parametrized by sub-session identifiers ssid. In $\mathcal{F}_{\mathrm{DLC}}$ and BC, ssids are encoded by labels of variables.

$\mathcal{F}_{\mathbf{BC}}$ Terminating Reliable Broadcast, with possibly external senders and receivers, also known as Byzantine Agreement, is formalized in Fig. 1.

$\mathcal{F}_{\mathrm{BC}}^{S,\mathcal{R}}$

If S corrupt, output $(\mathsf{network}, \mathsf{ssid})$ to $\mathcal{A}$. On input $\{$ $(\mathsf{input}, \mathsf{ssid}, v \in \{0,1\}^*)$ from S $\}$ OR $\{$only if S corrupt, $(\mathsf{activate}, \mathsf{ssid}, v \in \{0,1\}^* \sqcup \{\bot\})$ from $\mathcal{A}$ $\}$ then: if ssid is not stored, store it and delay-output (ssid, v) to every $R \in \mathcal{R}$.

Fig. 1: Terminating Reliable Broadcast for sender S and receivers $\mathcal{R}$.

$\mathcal{F}_{\mathbf{BC}}$ particularized to "Broadcast" BC and "Bulletin Board" bPKI. In the particular case of receivers $\mathscr{P}$, we denote $\mathsf{BC}^S := \mathcal{F}_{\mathrm{BC}}^{S,\mathscr{P}}$ for any $S \in \mathscr{P} \sqcup \mathscr{L}$. In the particular case of a sender $P \in \mathscr{P}$, we denote $\mathsf{bPKI}^P := \mathcal{F}_{\mathrm{BC}}^{P,\mathscr{P}\sqcup\mathscr{L}}$. Notice that the model of MPC denoted "PKI" [Dam+21] can be phrased as giving every player $P \in \mathscr{P}$ one access to bPKI^P prior to the start of the execution.

(Asynchronous) Authenticated Message Transmitting $\mathcal{F}_{\mathbf{AUTH}}$ as defined in [Can01], is formalized in Fig. 5 of §B. It is parametrized by a sender S and receiver R. On input v from S, it provides R with delayed output v.

Global Public Uniform Random String (URS) $\overline{\mathcal{G}}_{\mathbf{URS}}$. It samples uniformly at random a sequence of bits of pre-defined length κ, denoted URS, then outputs it to all players. It is further formalized in Figure 6 of §B, along with discussions on implementation. The following discussion is optional, as the reader may simply consider $\overline{\mathcal{G}}_{\mathrm{URS}}$ in the plain UC model. We strictly upgrade the security of our model in that we allow the string produced by $\overline{\mathcal{G}}_{\mathrm{URS}}$ to be directly observed by the Environment. In particular, our simulator will not have the choice but to use the URS provided by $\overline{\mathcal{G}}_{\mathrm{URS}}$ when simulating the contributions of honest players to the keys ($\mathbf{b}_i$ and $\mathbf{rlk}_i$). Furthermore, it will be clear from our UC proof in §5.4, i.e., Hybrid_2, based on Corollary 7, that the same URS can possibly re-used in multiple concurrent executions. Which, by the "EUC⇒GUC" in [CDPW02], implies that $\overline{\mathcal{G}}_{\mathrm{URS}}$ can be treated as a global resource.

Our Formalism of GOD. Consider a protocol Π, possibly in a hybrid model with ideal functionalities which may issue $\mathsf{network}$ queries to the adversary. Extending [Lam06], we denote as *complete* an execution of Π in which honest players took all the steps they could, and the adversary $\mathsf{activated}$ all $\mathsf{OutValReq}$ requests. We say that a protocol Π has GOD if, in every *complete* execution, then every honest player outputs. [Although our MPC protocol has finite executions, this definition also applies to infinite ones [DLS88, Remark 2].]

Claim: *If Π has GOD in the ($\overline{\mathcal{G}}_{URS}$,$\mathcal{F}_{AUTH}$,$\mathsf{BC}$,$\mathsf{bPKI}$) model, then, when ($\mathsf{BC}$, bPKI, $\mathcal{F}_{AUTH}$) are instantiated by protocols with guaranteed eventual output delivery, then Π has GOD in the usual sense.*

Proof: From the specifications of the previous functionalities, an execution of Π is complete if and only if: (i) all instances of BC and bPKI delivered an output to every player, whatever the sender, and (ii) for every input to $\mathcal{F}_{\mathrm{AUTH}}$ from a honest sender, then it was delivered to the receiver. Thus, when (i) $\mathcal{F}_{\mathrm{BC}}$ and bPKI are instantiated by Terminating Reliable Broadcast protocols, and (ii) $\mathcal{F}_{\mathrm{AUTH}}$ by channels with eventual delivery, then every execution of Π is complete.

2.3 UC: Ideal Functionality of MPC $\mathcal{F}_{\boldsymbol{C}}$.

Now that the issue of GOD is settled, we can focus on security in the UC sense, without anymore termination

considerations. The ideal functionality that we aim to UC implement, cf. §B.8 for reminders, is formalized as $\mathcal{F}_C$ in Fig. 2. It computes a public circuit C consisting of addition and multiplication gates in R_k. For simplicity: C has $|\mathscr{L}|$ input gates, one single output gate, $\mathcal{F}_C$ receives exactly one single input from each input owner, and delivers the output to all players and only them. For simplicity, C is hardcoded in $\mathcal{F}_C$. Inputs of owners which come as $\perp$ are arbitrarily set to 0. But our protocol $\Pi_C^{\mathcal{F}_{\text{DLC}}}$ allows straightforward generalizations that remove all these limitations, in particular, letting players adapt C based on the list of non-$\perp$ inputs received: see §B.7. This is actually what will do $\mathcal{F}_{\text{DLC}}$, for the case of linear combinations.

$\mathcal{F}_C$

Input For every corrupt owner ℓ, output $(\mathsf{network}, \ell)$ to $\mathcal{A}$. On input $\big\{(\mathsf{input}, m_\ell \in R_k)$ from any $\ell \in \mathscr{L}\big\}$ OR $\big\{$for any ℓ corrupt, on input $(\mathsf{activate}, \ell, m_\ell \in R_k \cup \{\perp\})$ from $\mathcal{A}\ \big\}$, if no value is stored for ℓ, then: (i) send to each player the delayed output $(\mathsf{input}, \ell, \overline{m_\ell})$; (ii) if $m_\ell = \perp$ then set $m_\ell := 0 \in R_k$; (iii) store $(\mathsf{input}, \ell, m_\ell)$.

Circuit evaluation Wait until for all $\ell \in \mathscr{L}$, there is some $(\mathsf{input}, \ell, m_\ell \in R_k)$ which is stored. Compute $y := C((m_\ell)_{\ell \in \mathscr{L}})$. Send to each player the delayed output m.

Fig. 2: Functionality of secure circuit evaluation.

2.4 Semi-Malicious corruptions. In our protocols and proofs we will consider what we define as *Semi-Malicious Corruptions*, following [Ash+12, §A.2] [BHP17; GLS15; BJMS20]. Semi-maliciously corrupt players and owners continuously forward to $\mathcal{A}$ their outputs received from ideal functionalities, and act arbitrarily as instructed by $\mathcal{A}$. E.g., they can possibly not send some message although the protocol instructs them to. However, when a corrupt entity M inputs a message m to $\mathcal{F}_{\text{AUTH}}$ or BC, then the sending of m must be compatible with the requirements of the protocol, with respect to: (i) all outputs of instances of $\overline{\mathcal{G}}_{\text{URS}}$, bPKI, BC, and also $\mathcal{F}_{\text{DLC}}$ in the case of our Π_C, required for sending m (ii) an internal witness tape that M must have, of the form (x, r) with x an input and r of the same length as all random coins that a honest player would have been meant to have tossed upon sending m. M can however use conflicting (x, r) when sending different messages m, m'. Finally, we also require that the semi-malicious adversary can only $\mathsf{activate}$ an output v to BC^M for some corrupt M only if: either v could have been input to BC^M by M itself according to the above rule, or, if $v = \perp$. Notice that we *do not* impose any condition for the sending of some m on bPKI. Concretely, this is because in the UC proof of Prop. §3 of our implementation of $\mathcal{F}_{\text{DLC}}$, we do not need extractable secret keys. See §5.6 for the favorable consequence of this relaxation.

3 $\mathcal{F}_{\mathbf{DLC}}$: Delayed Linear Combination over Rings R_q

The goal of this section is to specify then implement a functionality denoted $\mathcal{F}_{\mathrm{DLC}}$, specified in Figure 3. It is parametrized by a list of entities, denoted $\mathscr{S}$ the Senders, and for each of them by a predetermined list X_S of input labels $(\overline{x_{S,\alpha}})_{\alpha\in X_S}$. Here, all senders can be corrupt. In our MPC use-case, $\mathscr{S} := \mathscr{L}\sqcup\mathscr{P}$. Upon receiving all these inputs, in R_q, from Senders, $\mathcal{F}_{\mathrm{DLC}}$ accepts requests from players to open to them the evaluations of any linear form(s) over R_q on these inputs, of their choice. As specified, $\mathcal{F}_{\mathrm{DLC}}$ must wait to receive corrupt inputs (either from corrupt senders or directly from $\mathcal{A}$), and the deliveries of evaluations are arbitrarily delayed by $\mathcal{A}$. However it will be clear that our implementation Π_{DLC} has GOD when bPKI and $\mathcal{F}_{\mathrm{BC}}$ are implemented with terminating reliable broadcast, and $\mathcal{F}_{\mathrm{AUTH}}$ with eventual delivery.

$\mathcal{F}_{\mathrm{DLC}}$

Setup For each corrupt $P \in \mathscr{P}$: send $(\mathsf{network}, P)$ to $\mathcal{A}$. Upon receiving {(Setup) from any $P \in \mathscr{P}$} OR {if P corrupt, upon receiving $(\mathsf{activate}, \mathsf{Setup}, P)$ from $\mathcal{A}$ }, then: if no (Setup, P) is stored yet, then store it and provide every player with delayed output (Setup, P).
Wait until (Setup, P) is stored $\forall P \in \mathscr{P}$, then send ready to every $\mathscr{S}$.

Input For each corrupt $S \in \mathscr{S}$ and $\alpha \in X_S$: send $(\mathsf{network}, \overline{x_{S,\alpha}})$ to $\mathcal{A}$.
Upon receiving {$(\mathsf{input}, \overline{x_{S,\alpha}}, x_{S,\alpha} \in R_q)$, from any $S \in \mathscr{S}$} OR {for some corrupt S: $(\mathsf{activate}, \overline{x_{S,\alpha}}, x_{S,\alpha} \in R_q \cup \perp)$ from $\mathcal{A}$ }, then: (i) delay-output to every $P \in \mathscr{P}$: $(\mathsf{stored}, \overline{x_{S,\alpha}})$ appended with "$x_{S,\alpha} = \perp$" when the case; (ii) if $x_{S,\alpha} = \perp$ then set it to 0; (iii) store $(\mathsf{input}, S, x_{S,\alpha})$.

LCOpen Upon receiving input $(\mathsf{LCOpen}, \mathsf{ssid} = \Lambda)$ from $t+1$ players, such that all $\overline{x_{S,\alpha}}$ appearing with nonzero coefficient in Λ are stored, then delay output $(\mathsf{ssid} = \Lambda, y := \Lambda((x_{S,\alpha})_{S,\alpha})$ to every player.

Fig. 3: Delayed linear combination functionality. sid omitted

The high level implementation is as follows: to encrypt a secret s, generate a secret sharing of s in R_q with N shares and threshold t. Then for all $i \in [N]$, encrypt each share i under P_i's public key: this constitutes a Public Secret Sharing. Then to compute a linear combination Λ of some inputs (s_i): each player P decrypts its share $s_i^{(P)}$ of each s_i, then evaluates Λ on these shares: this constitutes a Partial Opening share. Then from any $t+1$ partial opening shares, the desired linear combination $\Lambda((s_i))$ is efficiently reconstructible. To make this idea work, we construct in §3.1 a secret sharing scheme which is *linear* over R_q, i.e., such that linear combinations on shares commute with linear combinations on secrets. Then, for the purpose of the UC proof of our implementation, in §3.2, we enrich our secret sharing scheme with elementary simulation gadgets which were not described explicitely, to our knowledge, in previous works on secret sharing over Galois rings.

3.1 Linear Secret Sharing over R_q. For any commutative ring A with unit 1, we denote as $A[Y]_t$ the polynomials of degree $\leq t$. For any $s \in A$, we denote $A[Y]_t^{(s)}$ the affine subspace evaluating at 0 to s, i.e., with constant coefficient s. Suppose that we are given a sequence $\{\alpha_0 := 0, \ldots, \alpha_N\} \in A^{N+1}$, such that all pairwise differences $\alpha_i - \alpha_j$ for $i \neq j$ are invertible in A. Such a sequence is denoted as "exceptional" in [Abs+19].

For $\mathcal{U} \subset (\alpha_i)_{i\in[0,\ldots,N]}$ we denote $\mathsf{Eval}_{\mathcal{U}} : P \in A[Y]_t \to [P(\alpha_i), i \in \mathcal{U}]$ the map returning the evaluations at points of $\mathcal{U}$. By [Abs+19, Thm 3], for every $(t+1)$-sized $\mathcal{U} \subset [0, \ldots, N]$, we have that $\mathsf{Eval}_{\mathcal{U}}$ is an *isomorphism*. For such $\mathcal{U}$, we denote the inverse of $\mathsf{Eval}_{\mathcal{U}}$ as "Polynomial Reconstruction" $\mathsf{PolReco}^{\mathcal{U}} : A^{t+1} \longrightarrow A[Y]_t$. We can construct is as: for each $i \in \mathcal{U}$, define the Lagrange polynomial as $\lambda_i^{\mathcal{U}}(Y) := \prod_{j\in\mathcal{U}\setminus\{i\}} \frac{Y-j}{i-j}$; then $\mathsf{PolReco}^{\mathcal{U}}([s^{(i)},\ i \in \mathcal{U}]) := \sum_{i\in\mathcal{U}} s^{(i)}.\lambda_i^{\mathcal{U}}(Y)$.

All what follows results from this isomorphism. First, we have the following [Abs+19, Construction 1] of *Linear Sharing over A*. Define the randomized function $\mathsf{LS}_A.\mathsf{Share} : A \longrightarrow A^N$, as: on input a secret $s \in A$, sample $P \leftarrow U\big(A[Y]_t^{(s)}\big)$ then return $\mathsf{Eval}_{(\alpha_i)_{i\in[N]}}(P) \in A^N$ denoted "shares" of s. We denote "(N,t)-secret sharing of s" any such possible output of $\mathsf{LS}_A.\mathsf{Share}(s)$. In the other direction, for a $(t+1)$-sized $\mathcal{H} \subset [N]$, we have the map $\mathsf{SReco}^{\mathcal{H}} := \mathsf{Eval}_{\{0\}} \circ \mathsf{PolReco}^{\mathcal{H}}$ which takes as input any $t+1$ shares and reconstructs the secret. Also, surjectivity of $\mathsf{Eval}_{\{0\}\cup\mathcal{I}}$ for any t-sized $\mathcal{I}$, implies Uniformity of any t shares of any fixed secret, cf. Property 16 of §C. In addition, let us introduce two useful A-linear maps:

- ***Perfect Simulation of (honest) Shares.*** For any t-sized $\mathcal{I} \subset [N]$, denote $\mathsf{ShSim}^{\mathcal{I}} := \mathsf{Eval}_{[N]\setminus\mathcal{I}} \circ \mathsf{PolReco}^{\{0\}\sqcup\mathcal{I}}$. It takes as input any t (typically corrupt) shares $(s^{(i)})_{i\in\mathcal{I}}$ with some secret s, and reconstructs the *unique* set of shares $(s^{(h)})_{h\in\mathcal{H}}$, with indices $\mathcal{H} := [N]\setminus\mathcal{I}$, such that the whole $(s^{(j)})_{j\in[N]}$ forms a (N,t)-secret sharing of s;
- ***Inference of (Corrupt) Shares.*** For any t+1-sized $\mathcal{H} \subset [N]$, denote $\mathsf{ShInfer}^{\mathcal{H}} := \mathsf{Eval}_{[N]\setminus\mathcal{H}} \circ \mathsf{PolReco}^{\mathcal{H}}$. It takes as input any $t+1$ shares $(s^{(h)})_{h\in\mathcal{H}}$, and reconstructs the *unique* set of shares $(s^{(i)})_{i\in\mathcal{I}}$, with (corrupt) indices $\mathcal{I} := [N]\setminus\mathcal{H}$, such that the whole $(s^{(j)})_{j\in[N]}$ forms a (N,t)-secret sharing of some s.

Construction of an Exceptional Sequence for R_q. The easy case is when all prime factors of q are of size at least $N+1$. Then we have that $[0, \ldots, N] \subset R_q$ forms an exceptional sequence. Indeed, all $i-j$ for $\big\{(i,j) \in [0,\ldots,N]^2,\ i \neq j\big\}$ are invertible modulo all the prime factors of q, thus are invertible modulo q by the Chinese remainders theorem (CRT), and thus in R_q. In the general case, we need to enlarge R_q. We do the construction for $q = p^e$ a prime power, then the case of composite q follows from the CRT. Denote $d := \lceil \log_p(N+1) \rceil$. The construction is conceptually as follows, and made explicit in full details in §C.1. Consider the $\mathbb{Z}/q\mathbb{Z}$-algebra $B := (\mathbb{Z}/q\mathbb{Z})[Y]/g(Y)$, where g is monic of degree d, and irreducible modulo p, denoted "Galois ring extension". In [Abs+19] they observe that B contains a p^d-sized exceptional sequence, from which they deduce linear secret-sharing over B, that we denote LS_B. By tensorisation of LS_B, over $\mathbb{Z}/q\mathbb{Z}$, with any inclusion of $\mathbb{Z}/q\mathbb{Z}$-algebras, e.g. $\mathbb{Z}/q\mathbb{Z} \hookrightarrow R_q$, we obtain a S-

linear secret-sharing scheme LS_S over $S := R_q \otimes_{\mathbb{Z}/q\mathbb{Z}} B$. R_q being a sub-ring of S, we have that LS_S particularizes to a R_q-linear sharing over R_q, denoted LS_{R_q}, as desired. Notice that, since each share is in $S \cong R_q^d$, we have a size overhead of d. But for simplicity, in the remaining we do as if shares were in R_q.

3.2 Implementation of $\mathcal{F}_{\mathbf{DLC}}$. Let $\mathsf{PKE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ be any public key encryption scheme satisfying IND-CPA. By convention, if the Enc of any plaintext fails, e.g., due to an incorrectly formatted public key $\mathsf{pk}^{\mathsf{PKE}}$, then it returns the plaintext itself.

Def-Prop 2 (Public Secret Sharing). *Let us consider the following randomized function* PSS*, parametrized by* N *strings* $(\mathsf{pk}_i^{\mathsf{PKE}})_{i\in[N]}$*. On input* $s \in R_q$*: compute* $(s^{(1)}, ..., s^{(N)}) \leftarrow \mathsf{LS}_{R_q}.\mathsf{Share}(s)$*, output* $\left[\mathsf{Enc}(\mathsf{pk}_i^{\mathsf{PKE}}, s^{(i)}), i \in [N]\right]$*.* PSS *is IND-CPA for any* $\mathcal{A}$ *being given at most* t *secret keys* $\left(\mathsf{sk}_i^{\mathsf{PKE}}\right)_{i\in\mathcal{I}\subset[N]}$*.*

An intuition of proof is that, under the idealized assumption that PKE ciphertexts under the unknown $t+1$ public keys would perfectly hide their content, then, the view of the adversary is the vector of t plaintext shares $[s^{(i)},\ i \in \mathcal{I}]$. But by Property 16 in §C, for any chosen plaintext s, it varies in $U(R_q^t)$. We carry out this idea in §C, by lack of reference, with a reduction to multi-messages multi-keys IND-CPA of PKE.

Π_{DLC}

Π_{DLC}**.Setup** $\forall P \in \mathscr{P}$: $(\mathsf{sk}_P^{\mathsf{PKE}}, \mathsf{pk}_P^{\mathsf{PKE}}) \leftarrow \mathsf{KeyGen}$, publish $\mathsf{pk}_P^{\mathsf{PKE}}$ on bPKI^P.

Π_{DLC}**.Input** Each sender $S \in \mathscr{S}$, upon output of all instances of bPKI^P $\forall P \in \mathscr{P}$, set $\mathsf{pk}_P^{\mathsf{PKE}}$ as the output of bPKI^P, possibly $\perp$. For each $\alpha \in X_S$:

(i) Compute $\boldsymbol{pss_{S,\alpha}} := \mathsf{PSS}\big((\mathsf{pk}_P^{\mathsf{PKE}})_{P\in\mathscr{P}},\ x_{S,\alpha}\big)$.

(ii) Broadcast $(\mathsf{input}, \mathsf{ssid} := \overline{x_{S,\alpha}},\ \boldsymbol{pss_{S,\alpha}})$ over BC^S.

Π_{DLC}**.LCOpen(Λ)** (i) $\forall P_j \in \mathscr{P}$, upon receiving output from all sub-instances of all BC^S whose label $\mathsf{ssid} = \overline{x_{S,\alpha}}$ has nonzero coefficient in Λ: for each output $(\overline{x_{S,\alpha}}, *)$, if $* = \perp$ then set $x^{(j)} := 0$; else if $* = [c_{S,\alpha}^{(1)}, \ldots, c_{S,\alpha}^{(n)}]$ then set $x_{S,\alpha}^{(j)} := \mathsf{PKE.Dec}(\mathsf{sk}_j, c_{S,\alpha}^{(j)})$. Evaluate $\mu^{(j)} := \Lambda\big((x_{S,\alpha}^{(j)})_{S,\alpha}\big)$ and send it over $\mathcal{F}_{\mathrm{AUTH}}^{P_j,R}$ to every player $R \in \mathscr{P}$.

(ii) Upon receiving opening shares $(\mu^{(i)})_{i\in\mathcal{U}}$ from any $(t+1)$-set $\mathcal{U} \subset [N]$ of players, outputs $\mu := \mathsf{LS}_{R_q}.\mathsf{SReco}^{\mathcal{U}}\big([\mu^{(i)},\ i \in \mathcal{U}]\big)$.

Fig. 4: Protocol for delayed linear combination

Proposition 3. *Protocol* Π_{DLC} *UC implements* $\mathcal{F}_{DLC}$

Proof. ***- Game 1.*** We modify the Real execution in that the opening shares of μ sent by honest players, are replaced as follows. Firstly, we define quantities denoted *Infered Corrupt Opening Shares* $(\mu^{(i)})_{i\in\mathcal{I}}$, nonwithstanding corrupt players may not have any opening shares on their witness tapes, since they may

not send any. For every input $x_{S,\alpha}$ of some honest S, we simply define $\big(x^{(i)}_{S,\alpha}\big)_{i\in\mathcal{I}}$ as the actual shares produced by S when it computes the PSS of $x_{S,\alpha}$. For each output $(\overline{x_{S,\alpha}}, *)$ of BC^S from some corrupt S: (i) if $* = \perp$ then we define $\big(x^{(i)}_{S,\alpha} := 0\big)_{i\in\mathcal{I}}$, otherwise (ii) this implies that $*$ is a correctly formed PSS. Thus in this case, we define as $\big(x^{(i)}_{S,\alpha}\big)_{i\in\mathcal{I}}$ the plaintext shares read on the witness tape of S. Finally, for all $i \in \mathcal{I}$ we set $\mu^{(i)} := \Lambda\big((x^{(i)}_{S,\alpha})_{\alpha\in X_S, S\in\mathscr{S}}\big)$. By linearity of LS_{R_q}, they are equal to the opening shares of μ that the $(P_i)_{i\in\mathcal{I}}$ would have sent if they were honest.

We now replace the honest opening shares by $\mathsf{ShSim}^{\mathcal{I}}\big(\mu, (\mu^{(i)})_{i\in\mathcal{I}}\big)$. Since $\mathsf{ShSim}^{\mathcal{I}}$ simulates perfectly, they are identical to the ones of the Real execution.

- *Game 2.* All the inputs $x_{\ell,\alpha}$ of honest $S \in \mathscr{S}$ are replaced by 0. Since the secret decryption keys sk_h of all honest players $h \in \mathcal{H}$ are not used anymore, we have the IND-CPA property of PSS which applies, thus the view of $\mathscr{E}$ is indistinguishable from the one in the previous game.

- *Game 3.* We now modify the method to *Infer* the corrupt shares of the $\boldsymbol{pss_{\ell,\alpha}}$ broadcast by corrupt senders ℓ. First, decrypt the honest shares of $\boldsymbol{pss_{\ell,\alpha}}$ using, again, the honest secret keys $(\mathsf{sk}_h)_{h\in\mathcal{H}}$. From them, infer the corrupt shares using $\mathsf{ShInfer}^{\mathcal{H}}$. The infered shares are identical to the ones in the previous game, by the property of $\mathsf{ShInfer}^{\mathcal{H}}$.

- *Game 4.* Honest players are now simulated, including generation of their their $(\mathsf{pk}_h, \mathsf{sk}_h)_{h\in\mathcal{H}}$, as well as all other honest senders, i.e., in $\mathscr{L}$. Their behavior is the same as in the previous game, thus both views have the same distribution.

But, in Game 4, the view is produced only from what is received from $\mathcal{F}_{\mathrm{DLC}}$ and from the adversary, without reading on the tapes of corrupt entities, i.e. without rewinding. Thus we are describing a simulator in the Ideal execution. □

4 BFV, with Relinearization from [CDKS19].

In §4.1 we recall RLWE and set some notations. In §4.2 we remind and specify the public key encryption scheme known as Brakerski-Fan-Vercauteren "BFV" [FV12]. To enable homomorphic multiplication, one needs what is denoted as a Relinearisation key (**rlk**). In §4.3, we present an *alternative* **rlk**, to the one in [FV12]. This alternative is none other than the particular case, in the case of a single key, of the one introduced in [CDKS19] in their context of MFHE. Then, we state a cryptographic hardness Assumption 6 made in [CDKS19] and denoted as "circular security", which is similar to ones in other works [FV12; GHS12a; Chi+16; CDKS19]. Finally in §4.4, we describe the homomorphic (addition and) multiplication algorithm using this relinearization key, and prove its correctness, which comes as a particular case of [CDKS19].

We do not state nor prove any security properties, they will instead come as a byproduct of the proof of MPC in 5.4, which is a strictly harder setting, due to adversarial influence on the public key (Lemma 12), re-use of random public parameters (Corollary 7), public intermediary decryption (§5.2.2 and Lem. 19).

4.1 Further Notations and Cryptographic primitives

We denote and $\lceil . \rceil, \lfloor . \rfloor, \lfloor . \rceil$ the rounding to the next, previous, and nearest integer respectively, and $[.]_k$ the reduction of an integer modulo k into R_k. When applied to polynomials or vectors, these operations are performed coefficient-wise. We denote $\Delta = \lfloor q/k \rfloor$, which is the integer division of q by k. For any element $\widetilde{r} = \Sigma_{i=0}^{n-1} \widetilde{r_i} X^i \in R$, we define its infinity norm as $\|\widetilde{r}\| := \max_i |\widetilde{r_i}|$. For $r \in R_q$, let us consider the unique representative $\widetilde{r} = \Sigma_{i=0}^{n-1} \widetilde{r_i} X^i \in R$ such that $\widetilde{r_i} \in [-(q-1)/2, ..., (q-1)/2]$ for all i. Then we define $\|r\| := \|\widetilde{r}\|$. We denote vectors of some length l (see §4.1) in bold, e.g. $\mathbf{a}$. For such vector $\mathbf{r} = (r_1, \ldots, r_l) \in R_q^l$, we define $\|\mathbf{r}\| := \max_i |\widetilde{r_i}|$. Finally, $\otimes : R_q^2 \times R_q^2 \to R_q^3$ denotes the tensor product.

Ring Learning with Errors. Let Ψ_q and $\mathscr{X}_q$ be distributions over R_q. The *decisional*-Ring Learning with Errors (RLWE) [LPR13a] assumption with parameter $(R_q, \mathscr{X}_q, \Psi_q)$ can be stated as follows: for a fixed secret sample $s \leftarrow \mathscr{X}_q$, then any polynomially long sequence of samples in R_q^2 of the form $(a_i, b_i = s\, a_i + e_i)_i$, where $a_i \leftarrow U(R_q)$, and $e_i \leftarrow \Psi_q$, is computationally indistinguishable from a uniform random sequence of elements of R_q^2.

Gadget Decomposition. For later use in §4.3.1 and §5.5, let us define the widely used, e.g., [GSW13; CDKS19; GMP19], *gadget toolkit*:

1. Gadget vector: $\mathbf{g} = (g_0, g_1, ..., g_{l-1}) \in R_q^l$; and integers l and (small) B_g;
2. The gadget decomposition denoted $\mathbf{g}^{-1}(.)$: on input any $x \in R_q$, decomposes it into a vector $\mathbf{u} = (u_0, ..., u_{l-1}) \in R^l$ of (small) coordinates, i.e, $\|u_i\| \leq B_g$ for all $0 \leq i \leq l-1$, such that $\Sigma_{i=0}^{l-1} u_i . g_i = x \pmod q$.

4.2 BFV as Standalone Public-Key Encryption

Parameters and Notations We consider common public parameters $pp = (R_q, l, \mathscr{X}_q, R_k, \Psi_q, \mathscr{B}_{Enc,q})$ that will be further specified in our MPC context in §5.3. In particular $(R_q, \mathscr{X}_q, \Psi_q)$ verify the RLWE assumption. Motivated by computational optimizations discussed in [FV12], [CH18, §A], [CDKS19], we specify n as a power of two, $f := X^n + 1$ the $2n$-th cyclotomic polynomial, and thus q odd. According to [Alb+21, p. 2.1.5], this enables to sample Ψ_q as: univariate polynomials with coefficients following a discrete Gaussian distribution centered at zero, then reduced modulo q. From the variance σ^2 of the distribution, one can derive an integer B such that the norms of coefficients are lower than B with overwhelming probability. Similarly, we consider an encryption noise distribution $\mathscr{B}_{Enc,q}$ of variance σ_{enc}^2 such that for an element sampled in $\mathscr{B}_{Enc,q}$ the norms of coefficients are lower than some B_{Enc} with overwhelming probability. We specify $\mathscr{X}_q$ as in [FV12, p5], [CDKS19, §6] and [MTBH20], as: polynomials in $\mathbb{Z}[X]/(X^n+1)$ with coefficients varying uniformly in $\{-1, 0, 1\}$, then reduced modulo q. Moreover, we specify our algorithms as taking a parameter denoted $\mathbf{a} \in R_q^l$. The reason is that, although in BFV the $\mathbf{a}$ is sampled locally uniformly, in our MPC protocol instead, as well as in [CDKS19], $\mathbf{a}$ will be common and sampled by $\overline{\mathcal{G}}_{\text{URS}}$, to enable some form of additivity.

Definition 4 (BFV as standalone public key encryption).

- $BFV.\mathsf{KeyGen}(\mathbf{a} \in R_q^l)$*: sample* $sk \leftarrow \mathscr{X}_q$, $\mathbf{e}^{(\mathbf{pk})} \leftarrow \Psi_q^l$, *output* $\mathbf{b} := -\mathbf{a}\, sk + \mathbf{e}^{(\mathbf{pk})}$ *and* sk.
- $BFV.\mathsf{Enc}(\mathbf{b}, \mathbf{a}, m \in R_k)$*: sample* $e_0^{(Enc)} \leftarrow \mathscr{B}_{Enc,q}$, $e_1^{(Enc)} \leftarrow \Psi_q$, $u \leftarrow \mathscr{X}_q$, *parse* $a := \mathbf{a}[0]$ *and* $b := \mathbf{b}[0]$. *Define the pair of linear forms* $\Lambda_{Enc}^{b,a} := \big(\overline{\Delta m} + \overline{u}b + \overline{e_0^{(Enc)}}, \overline{u}\, a + \overline{e_1^{(Enc)}}\big)$. *Output* $ct \leftarrow \Lambda_{Enc}^{b,a}\big(\Delta m, u, e_0^{(Enc)}, e_1^{(Enc)}\big) \in R_q^2$.
- $m \leftarrow BFV.\mathsf{Decrypt}(ct, sk)$*: Given a ciphertext* $ct = (ct[0], ct[1]) \in R_q^2$, *output* $m := [\left\lfloor \frac{k}{q}(ct[0] + ct[1].sk) \right\rceil]_k \in R_k$

Def-Prop 5 (Decryption noise).

Let $ct = (ct[0], ct[1]) \in R_q^2$, $m \in R_k$ *and* $sk \in R_q$. *We define the "decryption noise"* $e^{(Dec)}(ct, sk, m) := ct[0] + ct[1]\, sk - \Delta m$. *It satisfies the trivial property that if* $e^{(Dec)}(ct, sk, m) < \frac{\Delta}{2}$, *then,* $BFV.\mathsf{Decrypt}(ct, sk) = m$.

See in particular §D.2 for an upper-bound on the decryption noise of a freshly encrypted ciphertext.

4.3 Relinearization

Homomorphic multiplication involves two steps: the first, denoted "tensoring" produces a *degree two* ciphertext consisting of three elements $\hat{ct} = \left\lfloor \frac{k}{q} ct_1 \otimes ct_2 \right\rceil = (\hat{ct}[0], \hat{ct}[1], \hat{ct}[2]) \in R_q^3$. To reduce the degree back to one, a second step, denoted *relinearization*, must be carried out to turn $\hat{ct}$ into a size 2 ciphertext $ct' = (ct^{'}[0], ct^{'}[1])$ which can be decrypted as the product of the plaintexts. The latter requires what is denoted as a *relinearization key* (**rlk**). In the original BFV scheme [FV12], **rlk** takes the form of an encryption of the square of the secret key. In our context of secret-shared key, computing this squaring in an MPC manner would take 2 consecutive broadcasts, which is done in other works [Ash+12; MTBH20; Kim+20] but which we want to avoid. Fortunately, [CDKS19] introduced a new **rlk** for BFV which is *linear* in the secret key. Their context, denoted *multi*-key FHE (MFHE), is a priori different, since each player P generates its own secret key, from which it generates then publishes its own **rlk**, the global relinearization key then consisting in the vector of all $\mathbf{rlk}_i$. Our simple but seemingly new observation is that their method can be particularized to the single key case, and thus, applied to our TFHE setting. In this particularization, their vector collapses into a single **rlk**.

Notice that our description of **rlk** generation takes as argument some $\mathbf{d}_1 \in R_q^l$, which was instead sampled locally in [CDKS19]. The reason is that, in our MPC protocol, players will need to generate **rlk** distributively, in particular provide additive contributions denoted $(\mathbf{d}_{0,i}, \mathbf{d}_{2,i})$. But to enable additivity between these, they will need to be generated using the same common $\mathbf{d}_1$.

4.3.1 Relinearization Key Generation, Adapted from [CDKS19]

$(\mathbf{d}_0, \mathbf{d}_2) \in R_q^{l\times 2} \leftarrow \mathsf{CDKS}(\mathbf{a}, \mathbf{d}_1, sk)$: Given $sk \in R_q$ and $(\mathbf{a}, \mathbf{d}_1) \in R_q^{l\times 2}$:

1. Sample $r \leftarrow \mathscr{X}_q$.
2. Sample $\mathbf{e}_0^{(\mathbf{rlk})} \leftarrow \Psi_q^l$, and set $\mathbf{d}_0 = -sk\,\mathbf{d}_1 + \mathbf{e}_0^{(\mathbf{rlk})} + r\,\mathbf{g}$
3. Sample $\mathbf{e}_2^{(\mathbf{rlk})} \leftarrow \Psi_q^l$ and set $\mathbf{d}_2 = r\,\mathbf{a} + \mathbf{e}_2^{(\mathbf{rlk})} + sk\,\mathbf{g}$

4.3.2 Relinearization Algorithm *Relin*, Adapted from [CDKS19, §3.3.2]

The following algorithm takes as input $\hat{ct} = (\hat{ct}[0], \hat{ct}[1], \hat{ct}[2]) \in R_q^3$, $\mathbf{rlk} = \big(\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2\big) \in (R_q^l)^3, \mathbf{b} \in R_q^l$, and outputs $ct' = (ct'[0], ct'[1]) \in R_q^2$.

1 $ct'[0] \leftarrow \hat{ct}[0]$
2 $ct'[1] \leftarrow \hat{ct}[1]$
3 $ct'[2] \leftarrow \big\langle \mathbf{g}^{-1}(\hat{ct}[2]), \mathbf{b} \big\rangle$
4 $(ct'[0], ct'[1]) \leftarrow (ct'[0], ct'[1]) + \mathbf{g}^{-1}(ct'[2])^{\langle\cdot\rangle}(\mathbf{d}_0, \mathbf{d}_1)$
5 $ct'[1] \leftarrow ct'[1] + \big\langle \mathbf{g}^{-1}(\hat{ct}[2]), \mathbf{d}_2 \big\rangle$

4.3.3 Circular Security Hardness Assumption of [CDKS19]. Let us first state a more concrete equivalent statement: consider an oracle $\mathscr{O}_{\mathcal{D}_0}$ which samples $\mathbf{a} \leftarrow U(R_q^l)$ then KeyGenerates one BFV key pair $(sk, \mathbf{b})$, then samples $\mathbf{d}_1 \leftarrow U(R_q^l)$, then, using $\mathsf{CDKS}(\mathbf{a}, \mathbf{d}_1, sk)$, computes from it one public relinearization key $\mathbf{rlk} = (\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2)$ then outputs the pair $(\mathbf{b}, \mathbf{rlk})$. Then, any adversary has negligible advantage in distinguishing this *single* output from a single sampling in $U(R_q^{l\times 5})$.

Assumption 6. *Define the distribution:*

$$\mathcal{D}_0 := \Big\{(\mathbf{b}, \mathbf{a}, \mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2) : (\mathbf{a}, \mathbf{d}_1) \leftarrow U(R_q^l)^2,\ sk \leftarrow \mathscr{X}_q, (\mathbf{e}^{(\mathbf{pk})}, \mathbf{e}_0^{(\mathbf{rlk})}, \mathbf{e}_2^{(\mathbf{rlk})}) \leftarrow (\Psi_q^l)^3,$$
$$r \leftarrow \mathscr{X}_q,\ \mathbf{b} := -\mathbf{a}\,sk + \mathbf{e}^{(\mathbf{pk})},\ \mathbf{d}_0 := -sk\,\mathbf{d}_1 + \mathbf{e}_0^{(\mathbf{rlk})} + r\,\mathbf{g},\ \mathbf{d}_2 := r\,\mathbf{a} + \mathbf{e}_2^{(\mathbf{rlk})} + sk\,\mathbf{g}\Big\}$$

Then the maximum distinguishing advantage $\mathrm{Adv}_{\mathcal{D}_0}^{\lambda}$ *between a* single sample *in* $\mathcal{D}_0$ *and in* $U(R_q^{l\times 5})$*, is* $negl(\lambda)$.

We detail in §D.5 how Assumption 6 shows up in [CDKS19] with their notations. Very briefly, they first define a RLWE-based symmetric encryption scheme denoted UniEnc, for which they state (p7) and prove (§B.1) indistinguishability from uniform randomness of any pair {BFV public key; encryption of some chosen plaintext encrypted with UniEnc using the BFV secret key}, then they make the circular security assumption that indistinguishability still holds if one replaces the chosen plaintext by the BFV secret key itself.

For our MPC use-case, let us show that Assumption 6 implies the following Corollary 7. We first state a more concrete equivalent statement, for later use in the proof of MPC. Consider a public sampling of a uniform string $(\mathbf{a}, \mathbf{d}_1) \in U(R_q^{l\times 2})$, e.g., by $\overline{\mathcal{G}}_{\mathrm{URS}}$. Consider a polynomial number M of independent machines, each of them generates a key pair $(sk_m, \mathbf{b}_m)$ by using BFV.KeyGen, all using the common public $\mathbf{a}$. Likewise, each machine m generates $[\mathbf{d}_{0,m}, \mathbf{d}_{2,m}] \leftarrow$

$\mathsf{CDKS}(\mathbf{a}, \mathbf{d}_1, sk_m)$. Then the collection of the public data issued by these machines $\{\mathbf{b}_m, \mathbf{d}_{0,m}, \mathbf{d}_{2,m}\}_{m\in[M]}$, jointly with the public $(\mathbf{a}, \mathbf{d}_1)$, is still indistinguishable from one sample in $U(R_q^{(l\times 3)M} \times R_q^{l\times 2})$.

Corollary 7 (Security with Common Public Randomness). *Consider:*
$\mathcal{D}_0^M := \Big\{ \{\mathbf{b}_m, \mathbf{d}_{0,m}, \mathbf{d}_{2,m}\}_{m\in M}, \mathbf{a}, \mathbf{d}_1 : (\mathbf{a}, \mathbf{d}_1) \leftarrow U(R_q^l)^2, \;\; \textit{and}$
$\forall m \in [M] : \; sk_m \leftarrow \mathscr{X}_q, \; (\mathbf{e}_m^{(\mathbf{pk})}, \mathbf{e}_{0,m}^{(\mathbf{rlk})}, \mathbf{e}_{2,m}^{(\mathbf{rlk})}) \leftarrow (\Psi_q^l)^3, r_m \leftarrow \mathscr{X}_q, \; \mathbf{b}_m := -\mathbf{a}\, sk_m + \mathbf{e}_m^{(\mathbf{pk})},$
$\mathbf{d}_{0,m} := -sk_m\, \mathbf{d}_1 + \mathbf{e}_{0,m}^{(\mathbf{rlk})} + r_m\, \mathbf{g}, \; \mathbf{d}_{2,m} := r_m\, \mathbf{a} + \mathbf{e}_{2,m}^{(\mathbf{rlk})} + sk_m\, \mathbf{g} \Big\}$

Then the maximum distinguishing advantage $\mathrm{Adv}^{\lambda}_{\mathcal{D}_0^M}$ between a single sample in $\mathcal{D}_0^M$ and in $U(R_q^{(l\times 3)M} \times R_q^{l\times 2})$, is bounded by $M\mathrm{Adv}^{\lambda}_{\mathcal{D}_0}$.

Proof. Consider a cascade of oracles $\mathscr{O}_0 := \mathscr{O}_{\mathcal{D}_0^M}, \mathscr{O}_1, \ldots, \mathscr{O}_M$ such that each $\mathscr{O}_i$ returns the first i components of $R_q^{(l\times 3)M}$ in $U(R_q^{(l\times 3)i})$ and the remaining ones as in $\mathcal{D}_0^M$. Then the distinguishing advantage between two consecutive $\mathscr{O}_i$ is at most $\mathrm{Adv}_{\mathcal{D}_0}$, as a straightforward reduction shows. □

4.4 Homomorphic Evaluation of a circuit

We can now augment Definition 4 with homomorphic operations.

- (**Addition**) $BFV.Add(ct_1, ct_2)$: Return $ct = ct_1 + ct_2 \; \in R_q^2$.
- (**Multiplication**) $BFV.Mult(ct_1, ct_2, \mathbf{rlk}, \mathbf{b})$: Compute $\hat{ct} = \left\lfloor \frac{k}{q} ct_1 \otimes ct_2 \right\rceil \in R_q^3$ and return $ct' \leftarrow Relin(\hat{ct}, \mathbf{rlk}, \mathbf{b})$ (cf §4.3.2).
- $BFV.\mathsf{Eval}(C, \mathbf{rlk}, \mathbf{b}, (ct_\ell \in R_q^2)_{\ell\in\mathscr{L}})$, for a circuit C with $|\mathscr{L}|$ input gates: return the evaluation obtained by applying $BFV.Add$ and $BFV.Mult$ gate by gate, with inputs the $(ct_\ell)_{\ell\in\mathscr{L}}$.

In §D.4, we upper-bound the additional noise introduced by $BFV.Mult$ as the sum of the bounds given by eqns (12) and (14). These bounds are obtained by particularizing the analysis of [CDKS19, §C.1] in the single key setting, and turning their variances into essential upper-bounds. From them we deduce:

Proposition 8 (Decryption noise of a product). *Consider two BFV ciphertexts ct_1 and ct_2 of m_1 and m_2 respectively under a key $\big(\mathbf{b} = -\mathbf{a}\, sk + \mathbf{e}^{(\mathbf{pk})}, \mathbf{a}\big) \in R_q^{2\times l}$, with decryption noises (Def-prop 5) denoted $e_i^{(Dec)} := ct_i[0] + ct_i[1]\, sk - \Delta m_i$, $i \in \{1, 2\}$. Consider any $(\mathbf{d}_0, \mathbf{d}_2) \leftarrow \mathsf{CDKS}(\mathbf{a}, \mathbf{d}_1, sk)$, $\mathbf{rlk} = (\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2)$, denote $ct' := Mult(ct_1, ct_2, \mathbf{rlk}, \mathbf{b})$, then $e^{(Dec)}(ct', sk, m_1 m_2)$ is dominated by $k.n^2.\|sk\|(\|e_1^{(Dec)}\| + \|e_2^{(Dec)}\|) + n^2.l.B_g.\|sk\|(\|\mathbf{e}^{(\mathbf{pk})}\| + \|\mathbf{e}_2^{(\mathbf{rlk})}\|)$.*

Let us illustrate Def-Prop 5 with the example that, if the decryption noises of the ciphertexts inputs of a multiplication, i.e., $e_1^{(Dec)}$ and $e_2^{(Dec)}$, are such that the (approximate) bound in Proposition 8 above is (significantly) lower than $\frac{\Delta}{2}$, then, the decryption of the output ct' is the product of the plaintexts $m_1 m_2$.

5 Ad-Hoc TFHE with Short Ciphertexts and MPC

In §5.1 we outline the approach. In §5.2 we define a set of algorithms in the $\mathcal{F}_{\text{DLC}}$-hybrid model, which we denote as "Ad Hoc Threshold FHE" (ATFHE), for use in the MPC protocol $\Pi_C^{\mathcal{F}_{\text{DLC}}}$. Although not necessary for $\Pi_C^{\mathcal{F}_{\text{DLC}}}$, since the latter is also in the $\mathcal{F}_{\text{DLC}}$ hybrid model, we make explicit in the appendix §H how ATFHE, when compiled with Π_{DLC}, yields a list of actual algorithms for: Ad-Hoc encryption, TFHE distributed key generation, then threshold transformation from Ad Hoc to short BFV TFHE ciphertexts. In §5.2.1 we provide a distributed relinearization key generation in one broadcast. Then, in §5.2.2, we detail how to build a distributed decryption for ATFHE. We do not define the security of ATFHE on its own. Instead, we will show that our scheme is secure directly in the more general context of our MPC protocol $\Pi_C^{\mathcal{F}_{\text{DLC}}}$ provided in §5.3. Namely, in §5.4 we prove that $\Pi_C^{\mathcal{F}_{\text{DLC}}}$ UC-implements the MPC functionality $\mathcal{F}_C$ in the semi malicious model, with details in §F, then in §5.6 we compile it into malicious security. Finally, in §5.5 we show how to do bootstrapping. We provide additional details for a practical estimation of the parameters in §E.

5.1 Overall Approach of MPC using Ad-Hoc Threshold FHE

To introduce ATFHE, we outline how its algorithms will be used in the MPC protocol:

(0) **Setup:** Players receive a uniform string. *In parallel*, they Setup $\mathcal{F}_{\text{DLC}}$.
(1) **Share &** (2) **Shrink:** Input owners run ATFHE.Enc to send (**Share**) their inputs to $\mathcal{F}_{\text{DLC}}$. *In parallel*, based on the uniform string, players generate additive contributions: (DKG.Contrib) to the threshold BFV private key, which they input to $\mathcal{F}_{\text{DLC}}$, and to the public key, which they broadcast; (DRlkG.Contrib and DNG.Contrib) and to some extra material specific of TFHE. Then players Transform (**Shrink**) the inputs shared to $\mathcal{F}_{\text{DLC}}$ into BFV ciphertexts under the threshold key.
(3) **Evaluation** & **Threshold decryption:** Player compute an homomorphic evaluation of the circuit on these ciphertexts, then call PartDec to obtain a noised threshold decryption of this evaluation, then cleared with FinDec.

5.2 Ad Hoc Threshold Homomorphic Encryption (ATFHE)

We consider the model and notations of §2 and §4. Let pp denote common parameters of the form $(R_q, l, \mathscr{X}_q, R_k, \Psi_q, \mathscr{B}_{Enc,q}, \mathscr{B}_{sm,q})$. The novelty is $\mathscr{B}_{sm,q}$, which is the distribution over R_q, defined as: coefficients are distributed according to a centered discrete Gaussian with variance σ_{sm}^2 over $\mathbb{Z}^n$, that are subsequently reduced mod q. The choice of parameters will be specified in §5.3.1.

- ATFHE.DKG.Setup(): Obtain common uniform strings $(\mathbf{a}, \mathbf{d}_1) \leftarrow \overline{\mathcal{G}}_{\text{URS}}$, with $\mathbf{a} \in R_q^l$ and $\mathbf{d}_1 \in R_q^l$.
- DLC.Setup(): Send (Setup) to $\mathcal{F}_{\text{DLC}}$.

- $\mathsf{ATFHE.Enc}(m \in R_k)$: Sample $u \leftarrow \mathscr{X}_q$ and $e_0^{(Enc)} \leftarrow \mathscr{B}_{Enc,q}$, $e_1^{(Enc)} \leftarrow \Psi_q$, and send $(\mathsf{input}, \{\overline{\Delta m}, \overline{u}, \overline{e_0^{(Enc)}}, \overline{e_1^{(Enc)}}\}, \{\Delta m, u, e_0^{(Enc)}, e_1^{(Enc)}\})$ to $\mathcal{F}_{\text{DLC}}$.
- $\mathsf{ATFHE.DKG.Contrib}(\mathbf{a})$ for P_i: compute $(sk_i, \mathbf{b_i}) \leftarrow BFV.\mathsf{KeyGen}(\mathbf{a})$, i.e.: sample $\mathbf{e}_i^{(\mathbf{pk})} \leftarrow \Psi_q^l$ and $sk_i \leftarrow \mathscr{X}_q$, then $\mathbf{b_i} := -sk_i\,\mathbf{a} + \mathbf{e}_i^{(\mathbf{pk})}$. Send $(\mathsf{input}, \overline{sk_i}, sk_i)$ to $\mathcal{F}_{\text{DLC}}$. Output $\mathbf{b}_i$, denoted "additive contribution to the secret key".
- $\mathsf{ATFHE.DKG.Combine}\big(\{\mathbf{b}_i \in R_q^l\}_{i \in S \subset [N]}\big)$: Output $\mathbf{b} = \Sigma_{i \in S}\mathbf{b_i}$.
- $\mathsf{ATFHE.Transform}((\overline{\Delta m}, \overline{u}, \overline{e_0^{(Enc)}}, \overline{e_1^{(Enc)}}), \mathbf{b}, \mathbf{a})$: Given labels $(\overline{\Delta m}, \overline{u}, \overline{e_0^{(Enc)}}, \overline{e_1^{(Enc)}})$, a key $\mathbf{b}$ and $\mathbf{a}$, parse $a := \mathbf{a}[0]$ and $b := \mathbf{b}[0]$ then send $\big(\mathsf{LCOpen}, \Lambda_{Enc}^{b,a}(\overline{\Delta m}, \overline{u}, \overline{e_0^{(Enc)}}, \overline{e_1^{(Enc)}})\big)$ to $\mathcal{F}_{\text{DLC}}$.

5.2.1 Relinearization The following pair of algorithms is used in the MPC protocol to produce a common relinearization key, as defined in §4.3.1, compatible with the common threshold key pair. There, the outputs of the first one are used as inputs of the second.

- $\mathsf{ATFHE.DRlkG.Contrib}(\mathbf{a}, \mathbf{d}_1, sk_i)$: Output $(\mathbf{d}_{0,i}, \mathbf{d}_{2,i}) \leftarrow \mathsf{CDKS}(\mathbf{a}, \mathbf{d}_1, sk_i)$.
- $\mathsf{ATFHE.DRlkG.Combine}\big((\mathbf{d}_{0,i}, \mathbf{d}_{2,i})_{i \in S}, \mathbf{d}_1\big) := \big(\sum_{i \in S}\mathbf{d}_{0,i}, \mathbf{d}_1, \sum_{i \in S}\mathbf{d}_{2,i}\big)$.

5.2.2 Distributed Decryption and Conditions for its Correctness Recall from Definition 4 that the BFV decryption algorithm can be seen as a two steps process: (1) a linear combination $(c_0 + c_1 sk)$, (2) followed by a rescaling and rounding. However the raw output of (1) is not simulatable from the final decryption (2), and it actually leaks information about the secret (threshold) decryption key. Thus Asharov et al. [Ash+12] introduced the technique of adding additional noise, denoted *Smudging Noise*, to the output of the linear combination (1) before it can be opened. To implement this, we take the approach of [GLS15]: we specify a Distributed Noise Generation, denoted DNG, in which each player P_i (among the nonaborting ones $S \subset \mathscr{P}$) generates an additive contribution to the smudging noise, denoted η_i, which it stores in $\mathcal{F}_{\text{DLC}}$. Then, to open a ciphertext, players open all at once the sum of the linear combination (1) added with this noise. Surprisingly, all other works [Ash+12; Bon+18; Kim+20] use a comparatively less efficient technique, where each player adds independently a noise to its partial decryption before broadcasting it. Since reconstruction of shared secrets involves multiplication by Lagrange coefficients, these noises were blown up by $O((N!)^2)$.

- $\mathsf{ATFHE.DNG.Contrib}()$, for P_i: Sample $\eta_i \leftarrow \mathscr{B}_{sm,q}$, send $(\mathsf{input}, \overline{\eta_i}, \eta_i)$ to $\mathcal{F}_{\text{DLC}}$.
- $\mathsf{ATFHE.PartDec}(ct \in R_q^2, \{\overline{\eta}_i, \overline{sk_i}\}_{i \in S})$: Send $\big(\mathsf{LCOpen}, \Lambda_{Dec}^{ct}(\{\overline{\eta}_i, \overline{sk_i}\}_{i \in S})\big)$ to $\mathcal{F}_{\text{DLC}}$, where $\Lambda_{Dec}^{ct}(\{\overline{\eta}_i, \overline{sk_i}\}_{i \in S}) := ct[0] + ct[1].\Sigma_{i \in S}\overline{sk_i} + \Sigma_{i \in S}\overline{\eta_i}$.
- $\mathsf{ATFHE.FinDec}(\mu)$: Given $\mu \in R_q$, return $\big[\lfloor (k/q).\mu \rceil\big]_k$.

5.2.3 Correctness of Threshold Decryption Semi-malicious adversaries are not required to sample correctly, they are only required to have values of samples on their witness tapes, which are within the essential bounds on norms of the respective distributions, i.e., 1, B, B_{Enc} and B_{sm}. Thus, the definitions and bounds which we now provide are worst cases over the following choices:

- $\mathbf{a}$ and $\mathbf{d}_1$, both in R_q^l;
- for all $i \in [N]$: sk_i and r_i, in R_q and both of norm ≤ 1; $\mathbf{e}_i^{(\mathbf{pk})}$, $\mathbf{e}_0^{(\mathbf{rlk})}$ and $\mathbf{e}_2^{(\mathbf{rlk})}$; in R_q^l and all of norms $\leq B$;
- deduce from these, for all $i \in [N]$ the additive contributions to encryption and relinearization keys, namely: $(sk_i, \mathbf{b}_i)$ and $(\mathbf{d}_{0,i}, \mathbf{d}_{2,i})$, computed with the formulas in DKG.Contrib, resp, DRlkG.Contrib;
- Sum them into the threshold key $(sk, \mathbf{b})$ and the relinearization key $\mathbf{rlk}$, i.e., as specified in DKG.Combine, resp, DRlkG.Combine.

We now formalize the set in which belong the outputs of Transform. For any $m \in R_k$, we denote as a "Fresh BFV Encryption of m via a Transform", any element of R_q^2 of the form: $ct = (\Delta m + u.\,\mathbf{b}[0] + e_0^{(Enc)}, u\,\mathbf{a}[0] + e_1^{(Enc)})$, where $\|u\| \leq 1$, $\left\|e_0^{(Enc)}\right\| \leq B_{Enc}$ and $\left\|e_1^{(Enc)}\right\| \leq B$. Let us denote $e^{(fresh)} := e^{(Dec)}(ct, sk, m) := ct[0] + ct[1]\,sk - \Delta m$ its decryption noise (Def-Prop 5). A straightforward adaptation of the bound in the single-key setting, done in §D.2 (multiply by N the bounds on sk and $\mathbf{e}^{(\mathbf{pk})}$), yields:

$$\left\|e^{(fresh)}\right\| \leq B_{fresh} := 2nNB + B_{Enc}. \tag{1}$$

We now formalize at which condition the PartDec then FinDec of a homomorphically evaluated ciphertext, does return the correctly evaluated plaintext.

Def-Prop 9 (Decryption noise of a circuit: B_C). *For any arithmetic circuit C over R_k, with input gates indexed by $\mathscr{L}$, we consider the largest norm, over the previous choices, and over the choices: of elements $(m_\ell \in R_k)_{\ell \in \mathscr{L}}$, and of arbitrary fresh BFV Encryptions of them via* Transform *(as above) $(ct_\ell)_{\ell \in \mathscr{L}}$; denoting $ct := BFV.\mathsf{Eval}(C, \mathbf{rlk}, \mathbf{b}, (ct_\ell)_\ell)$ and $y := C((m_\ell)_\ell)$; of: the decryption noise $e^{(Dec)}(ct,\ sk,\ y)$. From Def-Prop 5 and the definition of* PartDec*, it follows that, for any y and ct as above,* If:

$$B_C + N.B_{sm} < \frac{\Delta}{2} \tag{2}$$

Then: $\mathsf{FinDec}\big(ct[0] + ct[1].sk\big) = y$.

5.2.4 Estimation of Noise after Homomorphic evaluation of a Circuit Let us now outline an estimation of B_C. Recall from §4.3 that multiplication involves two steps that increase the decryption noise: a first one denoted "*tensoring*", followed by a "*relinearization*". Let us consider a circuit $C = R_k^{\mathscr{L}} \to R_k$ of multiplicative depth L and input ciphertexts $(ct_\ell)_{\ell \in \mathscr{L}}$, each of decryption noise bounded by B_{fresh} (eq. (1)). The noise introduced by evaluating C is dominated by the one introduced by multiplications rather than additions, unless the width

is much larger than L, which we do not assume in this estimation. Thus we neglect, comparatively, the impact of $|\mathscr{L}|$. Using Proposition 8, we estimate an upper bound on the decryption noise of the evaluated ciphertext as:

$$C_1^L.B_{fresh} + C_2\Sigma_{i=0}^{L-1}C_1^i \leq C_1^L.B_{fresh} + L.C_2.C_1^{L-1} \tag{3}$$

with $C_1 = 2.k.n^2.N$ and $C_2 = 2.n^2.l^2.N^2.B.B_g$.

5.3 MPC Protocol $\boldsymbol{\Pi}_{\boldsymbol{C}}^{\mathcal{F}_{\mathrm{DLC}}}$

5.3.1 Parameters. $(R_q, l, \mathscr{X}_q, \Psi_q)$ are chosen so that Assumption 6 is verified. Notice that this assumption implies RLWE. Moreover we require Equation (2); and: (the former will be used in Lemma 11, the latter in Lemma 12)

$$\frac{B_C}{N.B_{sm}} = \mathrm{negl}(\lambda) \;\text{ and }\; \frac{2nNB}{B_{Enc}} = \mathrm{negl}(\lambda)\ . \tag{4}$$

5.3.2 Protocol $\boldsymbol{\Pi}_{\boldsymbol{C}}^{\mathcal{F}_{\mathrm{DLC}}}$ in $(\mathcal{F}_{\mathbf{DLC}}, \mathsf{BC})$-hybrid model, with external resource $\overline{\mathcal{G}}_{\mathbf{URS}}$ for computing an arithmetic circuit $\boldsymbol{C}$ over $\boldsymbol{R_k}$. Events are gathered by the strict ordering relation, e.g., steps denoted (1) are performed by honest players after they did the ones denoted (0), but on the other hand, all the steps (1) can be done concurrently by all entities, in arbitrary order.

- **Setup.**
  (0) Each player P_i sends (Setup) to $\mathcal{F}_{\mathrm{DLC}}$
  (0) Players run $(\mathbf{a}, \mathbf{d}_1) \leftarrow \mathsf{ATFHE.DKG.Setup}()$.

- **Inputs distribution.** Upon ready from $\mathcal{F}_{\mathrm{DLC}}$, each owner $\ell \in \mathscr{L}$:
  (1) Runs $\mathsf{ATFHE.Enc}(m_\ell)$ then goes offline.

- **Distributed Keys Generation.** Upon ready from $\mathcal{F}_{\mathrm{DLC}}$, each player P_i:
  (1) Runs $\mathsf{ATFHE.DKG.Contrib}(\mathbf{a})$, let $\mathbf{b}_i$ be the output;
  (1) Runs $\mathsf{ATFHE.DNG.Contrib}()$;
  (1) $(\mathbf{d}_{0,i}, \mathbf{d}_{2,i}) \leftarrow \mathsf{ATFHE.DRlkG.Contrib}(\mathbf{a}, \mathbf{d}_1, sk_i)$. Sends $\mathbf{b}_i, (\mathbf{d}_{0,i}, \mathbf{d}_{2,i})$ over BC^{P_i}

  For all $\ell \in \mathscr{L}$, each player waits to receive $(\mathsf{stored}, \ell, *_\ell)$ from $\mathcal{F}_{\mathrm{DLC}}$ for all fours variables of ℓ's $\mathsf{ATFHE.Enc}$; then sets $S_{ct} \subset \mathscr{L}$ the ℓ's for which no $*_\ell = \perp$. For all $P \in \mathscr{P}$, each player waits to receive $(\mathsf{stored}, P, *_P)$ from $\mathcal{F}_{\mathrm{DLC}}$ for both instances in DKG and DNG, and an output from all instances of $(\mathsf{BC}^P)_{P\in\mathscr{P}}$; then sets $S \subset \mathscr{P}$ the set of players for which no instance returned $\perp$.

- **Transformation and Evaluation.** Each player P_i:
  (2) $\forall j \in S$, parses the outputs of BC^{P_j} as $\mathbf{b}_j$ and $(\mathbf{d}_{0,j}, \mathbf{d}_{2,j})$. Computes
  $\mathbf{b} \leftarrow \mathsf{ATFHE.DKG.Combine}(\{\mathbf{b}_j\}_{j\in S})$ and
  $\mathbf{rlk} \leftarrow \mathsf{ATFHE.DRlkG.Combine}((\mathbf{d}_{0,j}, \mathbf{d}_{2,j})_{j\in S}, \mathbf{d}_1)$.

(2) $\forall \ell \in S_{ct}$, runs $\mathsf{ATFHE.Transform}((\overline{\Delta m_\ell}, \overline{u_\ell}, \overline{e_{0,\ell}^{(Enc)}}, \overline{e_{1,\ell}^{(Enc)}}), \mathbf{b}, \mathbf{a})$.

(3) Upon receiving $(\Lambda_{Enc}^{a,b}, ct_j)$ from $\mathcal{F}_{\mathrm{DLC}}$ for all $j \in S_{ct}$, computes $ct \leftarrow BFV.\mathsf{Eval}(C', \{ct_j\}_{j\in S_{ct}}, \mathbf{rlk}, \mathbf{b})$, where C' is the circuit induced from C by setting $m_\ell := 0\ \forall \ell \notin S_{ct}$.

(3) Runs $\mathsf{ATFHE.PartDec}(ct, \{\overline{\eta_i}, \overline{sk_i}\}_{i\in S})$.

- **Output computation.** Each player P_i:
  - Upon receiving $(\Lambda_{Dec}^{ct},\ \mu)$ from $\mathcal{F}_{\mathrm{DLC}}$, outputs $m := \mathsf{ATFHE.FinDec}(\mu)$.

5.3.3 Communication complexity of $\Pi_C^{\mathcal{F}_{\mathrm{DLC}}\to\Pi_{\mathsf{DLC}}}$, i.e., when $\mathcal{F}_{\mathrm{DLC}}$ is instantiated with Π_{DLC}. Each player initially broadcasts PSSs of its key and noise shares, each of size $O(N\, n\, \log q)$ bits, as well as its public and relinearization keys of size $O(n\, \log q)$ bits. Thus overall, $O(N^2\, n\, \log q)$ bits are broadcast. In parallel, each input owner initially broadcasts an ad-hoc ciphertext of size $O(N\, n \log q)$ bits. After transformation, ciphertexts are of size $2.n \log q$ and, by construction of the BFV scheme, so are the evaluated ciphertexts and the partial decryptions.

5.4 UC Proof of Main Theorem 1

Theorem 10. *$\Pi_C^{\mathcal{F}_{DLC}}$ UC implements the ideal functionality $\mathcal{F}_C$ for any semi-malicious adversary, in the $(\mathcal{F}_{DLC}, \mathsf{BC})$-hybrid model with external resource $\overline{\mathcal{G}}_{URS}$.*

By Prop. 3, Π_{DLC} UC implements $\mathcal{F}_{\mathrm{DLC}}$. Thus, Thm. 10 implies Thm. 1. We now prove Thm 10 and give additional details in §F.

Description of the Simulator $\mathcal{S}$ of $\Pi_C^{\mathcal{F}_{DLC}}$. $\mathcal{S}$ Initiates in its head: sets of N players and $\mathscr{L}$ of Owners, and may initially receive corruption requests from $\mathscr{E}$ for arbitrarily many owners and up to t players, indexed by $\mathcal{I} \subset \mathscr{P}$. It simulates functionalities $\mathsf{BC}, \mathcal{F}_{\mathrm{DLC}}$ following a correct behavior, apart from the value returned by $\mathcal{F}_{\mathrm{DLC}}$ in the Output computation. For instance, receiving activate from $\mathscr{E}$ intended to some functionality, $\mathcal{S}$ internally sends it to the functionality then simulates the steps taken by the functionality accordingly. Upon every output from a simulated functionality to a simulated corrupt player, or, upon every message from a simulated functionality to $\mathcal{S}$ acting as $\mathcal{A}$, then $\mathcal{S}$ immediately forwards it to $\mathscr{E}$, as would have done the actual dummy $\mathcal{A}$.

- **Setup.**
  (0) Simulates the setup of $\mathcal{F}_{\mathrm{DLC}}$, i.e., queries network for corrupt players then, upon receiving Setup or activate on behalf of all corrupt players, sends delayed output ready to players and owners.

  (0) Retrieves $(\mathbf{a}, \mathbf{d}_1)$ from $\overline{\mathcal{G}}_{\mathrm{URS}}$ and sends it to all on behalf of $\overline{\mathcal{G}}_{\mathrm{URS}}$

- **Input distribution:** Simulates of a correct behavior of $\mathcal{F}_{\mathrm{DLC}}$. Moreover:
  (1) $\forall$ simulated honest $\ell \in \mathscr{L}$: sets $\widetilde{m_\ell} := 0$ and runs $\mathsf{ATFHE.Enc}(\widetilde{m_\ell})$.

(1) $\forall$ simulated corrupt $\ell \in \mathscr{L}$, upon $\Big(\mathsf{input}$ OR $\mathsf{activate}, \{\overline{\Delta m_\ell}, \overline{u_\ell}, \overline{e_{0,\ell}^{(Enc)}}, \overline{e_{1,\ell}^{(Enc)}}\}, \{\Delta m_\ell, u_\ell, e_{0,\ell}^{(Enc)}, e_{1,\ell}^{(Enc)}\}\Big)$ from $\mathscr{E}$, sets $\widetilde{m_\ell} := 0$ if $m_\ell = \perp$ or $\widetilde{m_\ell} := m_\ell$ otherwise, and sends $(\mathsf{input}, \ell, m_\ell)$ to $\mathcal{F}_C$.

- **Distributed Keys Generation:** Simulates of a correct behavior of $\mathcal{F}_{\text{DLC}}$. For every simulated honest $P_i \in \mathcal{H}$:
  (1) Samples $sk_i \leftarrow \mathscr{K}_q$ (never used) and $\eta_i \leftarrow \mathscr{B}_{sm,q}$, and sends them to $\mathcal{F}_{\text{DLC}}$.
  (1) Samples $\mathbf{b}_i \leftarrow U(R_q^l)$ and $(\mathbf{d}_{0,i}, \mathbf{d}_{2,i}) \leftarrow U(R_q^l \times R_q^l)$, sends them over BC^{P_i}.
  As in the protocol, $\mathcal{S}$ sets $S_{ct} \subset \mathscr{L}$ the owners, resp. $S \subset \mathscr{P}$ the players, for which no instance returned $\perp$.
- **Transformation and Evaluation** Simulates correct behaviors. For instance in (2), denoting $\mathbf{b} := \sum_{j\in S} \mathbf{b}_j$ and $\mathbf{rlk} := \mathsf{DRlkG.Combine}\big((\mathbf{d}_{0,j}, \mathbf{d}_{2,j})_{j\in S}, \mathbf{d}_1\big)$, the simulated $\mathcal{F}_{\text{DLC}}$, for $\forall \ell \in S_{ct}$, delay-outputs:
  $\big(\Lambda_{Enc}^{a,b}, \widetilde{ct_\ell} := (\Delta\widetilde{m_\ell} + u_\ell\, b + e_{0,\ell}^{(Enc)}, u_\ell\, a + e_{1,\ell}^{(Enc)})\big)$.
- **Output computation:** Upon receiving $(\mathsf{evaluation}, y)$ from $\mathcal{F}_C$, where by definition $y = C'(\{m_\ell\}_{\ell\in S_{ct}})$, then $\mathcal{S}$ simulates the following incorrect behavior:
  - $\mathcal{F}_{\text{DLC}}$ delay-outputs $\Big(\Lambda_{dec}^{ct},\ \mu^{\mathcal{S}} := \Delta y + \Sigma_{j\in S}\eta_j\Big)$.

Proof of indistinguishability with a real execution We go through a series of hybrid games, starting from the real execution $REAL_{\Pi_C}$. For completeness they are spelled out in full details in §F.3. The view of $\mathscr{E}$ consists of its interactions with $\mathcal{A}/\mathcal{S}$, and of the outputs of the actual honest players. We deal with the latter once and for all in Lemma 11.

Hybrid_1 ***[Simulated Decryption].*** $\mathcal{F}_{\text{DLC}}$ is modified in Output computation: there it, incorrectly, outputs $\mu^{\mathcal{S}} := \Delta.y + \Sigma_{j\in S}\eta_j$, where $y := C'((m_\ell)_{\ell\in S_{ct}})$ is the evaluation in clear of the circuit on the actual inputs.

Lemma 11. *The outputs of the actual honest players are the same in $REAL_{\Pi_C}$ and $IDEAL_{\mathcal{F}_C,\mathcal{S},\mathscr{E}}$. Also, the views of $\mathscr{E}$ in $REAL_{\Pi_C}$ and Hybrid_1 are computationally indistinguishable.*

Proof. It is convenient to prove the two claims at once. The view of $\mathscr{E}$ is identical in $REAL_{\Pi_C}$ and Hybrid_1 until (3) included. There, for all $\ell \in S_{ct}$, consecutively to $\mathsf{ATFHE.Enc}$ and $\mathsf{ATFHE.Transform}$, $\mathcal{F}_{\text{DLC}}$ delay-outputs a fresh BFV encryption ct_ℓ of m_ℓ via $\mathsf{Transform}$ under $(\mathbf{b}, \mathbf{a})$, following the terminology of §5.2.3. Thus, the evaluated $ct := BFV.\mathsf{Eval}(C', \{ct_j\}_{j\in S_{ct}}, \mathbf{rlk}, \mathbf{b})$ is the same in both views. In the Output computation of $REAL_{\Pi_C}$, the output of $\mathcal{F}_{\text{DLC}}$ is:

$$\mu = ct[0] + ct[1].\Sigma_{j\in S} sk_j + \Sigma_{j\in S}\eta_j\,, \tag{5}$$

with $\eta_j \leftarrow B_{sm}$ for all $j \in S$. First, by Def-Prop 9, we have, for some noise $e^{(Dec)}$, with $\|e^{(Dec)}\| \leq B_C$

$$ct[0] + ct[1].\Sigma_{j\in S} sk_j = \Delta y + e^{(Dec)}\,. \tag{6}$$

Since $\eta_j \leq B_{sm}$ for all $j \in S$, it follows from the choice of parameters (2) and the final remark in Def-Prop 9, that the output of honest players in $REAL_{\Pi_C}$ is $m := \mathsf{ATFHE.FinDec}(\mu) = y$, which proves our first claim. Second, since we specified $\|e^{(Dec)}\|/N.B_{sm} = negl(\lambda)$ (equation (4)), it follows that the distribution of μ, given by (5) is computationally indistinguishable from the one of $\Delta y + \Sigma_{j\in S}\eta_j$, see the "smudging" Lemma 19 in §F for a further formalization of this fact. But the latter is by definition $\mu^{\mathcal{S}}$, which is exactly the output of $\mathcal{F}_{\mathrm{DLC}}$ in Hybrid_1. □

Hybrid_2 ***[Random Public Keys].*** This is the same as Hybrid_1 except that the additive contributions $(\mathbf{b}_i, (\mathbf{d}_{0,i}, \mathbf{d}_{2,i}))_{i\in\mathcal{H}}$ of honest players to the public and relinearization keys, are replaced by a sample in $U(R_q^{l\times 3})$. Indistinguishability from Hybrid_1 follows from Corollary 7.

Hybrid_3 ***[Bogus Honest Inputs]*** This is the same as Hybrid_2 except that $\mathsf{ATFHE.Enc}$ on behalf of honest owners are computed with $\widetilde{m_\ell} := 0$, instead of with their actual inputs m_ℓ. Importantly, the behavior of $\mathcal{F}_{\mathrm{DLC}}$ is unchanged, i.e., correct until (3) included, then outputs $\mu^{\mathcal{S}} := \Delta y + \Sigma_{j\in S}\eta_j$, where $y := C'((m_\ell)_{\ell\in S_{ct}})$ is still the evaluation of the circuit on the *actual* inputs.

We now have that Hybrid_3 and $IDEAL_{\mathcal{F}_C,\mathcal{S},\mathscr{E}}$ produce identical views to $\mathscr{E}$. Indeed, the behaviours of $\overline{\mathcal{G}}_{\mathrm{URS}}$, of the simulated ideal functionalities $(\mathcal{F}_{\mathrm{DLC}}, \mathsf{BC})$, and of the honest parties in Hybrid_3, are identical to the simulation done by $\mathcal{S}$.

Lemma 12. Hybrid_2 *and* Hybrid_3 *are computationally indistinguishable.*

Proof. Since Hybrid_2, the secret keys of the honest players ($P_i \in \mathcal{H}$) are no longer used in any computation. Furthermore, since honest players sample their contributions $\mathbf{b}_i$ to the BFV public key independently (uniformly at random), we can assume without loss of generality that corrupt contributions are generated after having seen the honest ones. We can thus apply Lemma 18 "IND-CPA under Joint Keys", which adapts the one of [Ash+12, Lemma 3.4] in the RLWE setting. It considers a uniform value $\mathbf{b}$ in R_q, then the adversary can add to it the sum $\mathbf{b}'$ of t BFV public keys which it semi-maliciously produces (compatibly with $\mathbf{a}$). It states that the ciphertext of a chosen message under the sum of keys $\mathbf{b}+\mathbf{b}'$, is still indistinguishable from a uniformly random value. The reduction, from multi-message, to this latter single-message statement, is straightforward. □

5.5 In Addition: Enabling Bootstrapping

The "Bootstrapping" of a single-key BFV ciphertext is a local computation that homomorphically brings back the size of its decryption noise, roughly to the one of a fresh ciphertext. Our starting point is the one described in [CDKS19, §5] in their context of multikey BFV. As we did for relinearization, we particularize it to our single-key context. Leaving all details in §G, let us just mention that two elementary homomorphic operations in bootstrapping require an auxiliary key. The latter consists of a collection of keys, indexed by $j \in (\mathbb{Z}/2n)^*$, constructed as follows in the single key setting. Let sk be the decryption key,

considering τ_j over R_q, sample $\mathbf{h}_1 \leftarrow U(R_q^l)$ and $\mathbf{e}^{(\mathbf{bk})} \leftarrow \Psi_q^l$ and output $\big(\mathbf{h}_1,\ \mathbf{h}_0(j) = -sk\,\mathbf{h}_1 + \mathbf{e}^{(\mathbf{bk})} + \tau_j(sk)\,\mathbf{g}\big)$. We enrich the MPC protocol with generation of this bootstrapping key without changing the communication complexity, as follows. Receive $\mathbf{h}_1 \leftarrow U(R_q^l)$ from $\overline{\mathcal{G}}_{\mathrm{URS}}$. For each player i, let sk_i be its additive contribution to the secret key, used in the ATFHE.DKG.

- ATFHE.BkDKG.Contrib($\mathbf{h}_1, sk_i, j$): Given a secret key contribution sk_i, sample $\mathbf{e}_i^{(\mathbf{bk})} \leftarrow \Psi_q^l$ and compute $\mathbf{h}_{0,i}(j) = -sk_i\,\mathbf{h}_1 + \mathbf{e}_i^{(\mathbf{bk})} + \tau_j(sk_i)\,\mathbf{g}$.
- ATFHE.BkDKG.Combine($\mathbf{h}_1, \{\mathbf{h}_{0,i}(j)\}_{i\in S}, j$): Given a set $\{\mathbf{h}_{0,i}(j)\}_{i\in S}$, output $\mathbf{bk}(j) := (\Sigma_{i\in S}\mathbf{h}_{0,i}(j), \mathbf{h}_1)$

Remark 1. To add bootstrapping in the UC simulation, we must enlarge the assumption 6 in order to take into account these new keys. This is done implicitly in [CDKS19]. From such enlarged assumption, we deduce the analogous of Corollary 7, w.r.t. $\mathbf{h}_1 \leftarrow U(R_q^l)$.

5.6 Semi-malicious Security to Malicious Security

At a high level, malicious security can be achieved by applying the compiler of [Ash+12, §E], i.e., by instructing players and owners to ZK prove, upon sending messages, knowledge of a witness as in the semi-malicious model. But the compiler of [Ash+12] is designed for broadcast-based protocols, whereas in ours, players in (2) and (3) also act based on previous outputs of $\mathcal{F}_{\mathrm{DLC}}$. This is why we also required the x to contain these outputs, in our model §2.4. We can also simplify their compiler by allowing players to prove their statements with UC NIZKs, recalled in §B.9, since these can be set-up under honest majority from one initial call to bPKI, thanks to the technique denoted Multi-String CRS [GO14; BJMS20]. On the face of it, this call pre-pends one more preliminary non-Responsive event to the execution. However, we can actually have players publish multi-strings *in parallel* of the initial event, i.e., of their access to bPKI. Indeed in our semi-malicious model §2.4 we did not impose any condition when broadcasting over bPKI. Multi-Strings instead of $\overline{\mathcal{G}}_{\mathrm{URS}}$-based NIZKs have the merits (i) to relieve Owners from the need to access $\overline{\mathcal{G}}_{\mathrm{URS}}$ when constructing their NIZKs, and (ii) to preserve $\overline{\mathcal{G}}_{\mathrm{URS}}$ as a global resource, which would otherwise have needed to be simulated if used to produce NIZKs: see §B.9.

6 Impossibility of 1-Broadcast-then-Asynchronous MPC

"Secure channels" is the upgrade of $\mathcal{F}_{\mathrm{AUTH}}$ where network requests do not leak the content of messages to $\mathcal{A}$. The functionality of simultaneous broadcast, denoted SB, is parametrized by two senders P_1 and P_N, and returns to all players a pair (x_1, x_N) such that, for $i \in \{1, N\}$, x_i is the input of P_i if it is honest.

Theorem 13. *Consider a hybrid network in which players are* initially *given access to one single round of: synchronous pairwise secure channels and broadcast;* then: *only access to pairwise secure channels with guaranteed eventual delivery.*

Then for any $t \geq 3$ and $N \leq 3t-4$ there is no computationally secure SB *protocol.*

Proof. We assume such a protocol for $N = 3t-4$. We construct an adversary $\mathcal{A}$ which corrupts P_N but not P_1, and still manages so that the second output, x_N, will be *correlated* with the input of the honest P_1. However, in an ideal execution, then there is no link between the adversary and P_1 until the moment when SB reveals (x_1, x_N). Thus, the influence of $\mathcal{A}$ on the outcome is unachievable in the ideal execution, hence a contradiction.

We define the subsets $\mathcal{Q} := \{P_1, \ldots, P_{N-t=2t-4}\}$ and $\mathcal{Q} := \{P_t, \ldots, P_{N-1=3t-5}\}$, which we may denote as "quorums". The $3t-4$ comes from the following tight constraints: We managed to have $|\mathcal{Q}| = |\mathcal{Q}'| = N - t$, so that in our proof, each of these quorums, respectively, when all of them behave honestly and do not hear from the outside, then they must output in a noninfinite number of steps. $\mathcal{A}$ corrupts P_2. $\mathcal{A}$ corrupts a well-chosen player $P_i \in \mathcal{Q}$, which we will detail. $\mathcal{A}$ selects a player P_j in $\mathcal{Q}'$ at random and corrupts it. $\mathcal{A}$ corrupts the remaining players in $\mathcal{Q} \cap \mathcal{Q}'$, which it can do since they are at most $t-3$, reaching a total of at most t corruptions.

In the first round, all players act honestly, except P_N, which we will detail.

Then in the asynchronous phase, P_N is silent. $\mathcal{A}$ cuts the network in two, i.e.: $\mathcal{A}$ does not deliver {any message from honest players in $\mathcal{Q}$ to honest players in $\mathcal{Q}'$}, nor {any message from honest players in $\mathcal{Q}'$ to honest players in $\mathcal{Q}$}. $\mathcal{Q} \cap \mathcal{Q}'$ behaved so honestly so far, thus is meaningful to define the internal states that they would have if they were honest. $\mathcal{A}$ makes a copy of these $\mathsf{States}(\mathcal{Q} \cap \mathcal{Q}')$.

$\mathcal{A}$ first "probes" the value of x_1 as follows. It "freezes" $\mathcal{Q}$, i.e., does not deliver any message to honest players in $\mathcal{Q}$. $\mathcal{A}$ makes the corrupt players in $\mathcal{Q}'$ play honestly, and has messages in $\mathcal{Q}'$ delivered in round-robin order. Thus, the view of honest players in $\mathcal{Q}'$ is indistinguishable from an execution in which {$\mathcal{Q}'$ were the $n-t$ honest players, while the remaining ones would be silent}. Thus, they all output after some non-infinite number of steps. In particular, since P_j plays the correct algorithm, it will learn the same output as honest players. With probability $1/(N-t)$, P_j is the *first* in $\mathcal{Q}'$ which learns the output (x_1, x_N).

If the probing is successful, i.e., upon this event, then we have that (i) $\mathcal{A}$ *learns* the actual input value x_1 of P_1; (ii) whereas *no* honest player has output yet, thus, the *actual* output of the protocol is *not yet* defined. In this case, then $\mathcal{A}$ (immediately) "freezes" $\mathcal{Q}'$, i.e., delivers no more message to honest players in $\mathcal{Q}'$, so that none of them will ever output. Then $\mathcal{A}$ reinitializes corrupt players in $\mathcal{Q} \cap \mathcal{Q}'$ back to their $\mathsf{States}(\mathcal{Q} \cap \mathcal{Q}')$ just after the first round. Then $\mathcal{A}$ "un-freezes" the set of honest players in $\mathcal{Q}$, and have all players in $\mathcal{Q}$ play honestly the protocol, with their messages delivered in round-robin order. Let us now specify the behavior of P_i.

Choosing P_i and its joint action with $\mathcal{A}$. By (B, M), where $M = (M_1, \ldots, M_N)$ we denote joint distribution of broadcast and messages sent by P_N in round 1. By (B^0, M^0) and (B^1, M^1) we denote the *honest* distributions corresponding to inputs $x_N = 0$ and $x_N = 1$. For a distribution (B, M), and $\sigma \in \{0, 1\}$, let $q_\sigma(B, M)$ the probability that the protocol's output second entry

be 1, given that (1) $x_1 = \sigma$ (2) P_N's messages in round 1 are (B, M) then P_N becomes silent (3) all players in $\mathcal{Q}$ follow the protocol, with their messages delivered in round-robin order, while players in $\mathcal{Q}' \backslash \mathcal{Q}$ play honestly in round 1 then become silent. Finally, let $q(B, M) := (q_0(B, M), q_1(B, M))$

Lemma 1. *(B^0, M_1^0) and (B^1, M_1^1) are computationally indistinguishable.*

Proof. Assume that there exists a distinguisher $\mathcal{D}$ with non-negligible advantage. Then we could construct an adversary $\mathcal{A}_1$ corrupting P_1, but not P_N, rushing to learn (B^0, M_1^0) before all other players, submit it to $\mathcal{D}$, obtain an estimate of x_N, and choose P_1's input equal to this estimate, obtaining a non-negligible correlation. □

Corollary 14. $\forall \sigma \in \{0, 1\}, \quad q_\sigma(B^0, M_1^0) - q_\sigma(B^1, M_1^1) = \mathsf{negl}$.

Proof. For each σ, we build a distinguisher $\mathscr{E}$ for the distributions of Lemma 1 as follows. Simulate a set $\mathscr{P}$ of players, all starting with a blank state excepted P_1 with input σ. Upon receiving a challenge (B^b, M_1^b), make P_N broadcast B^b and send M_1^b, while the other players play the first round honestly. Then silence all players not in $\mathcal{Q}$, simulate an execution complete for $\mathcal{Q}$ with messages delivered in round robin order, and output the same $b := x_N$ as players. □

Remark. This is where we used that there is no setup. Otherwise, if there had been a PKI, then $\mathscr{E}$ would have had to initialize players with an internal state compatible with their public keys, thereby somehow guessing their secret keys. Actually, a PKI *does* enable a N-SB, in which players broadcast PVSSs of their inputs, then threshold-decrypt the PVSSs over asynchronous channels.

Consider now the following four pairs of probabilities:

$$\begin{aligned} Q_1 &= q(B^1, M_1^1, \ldots, M_{N-t}^1, 0, \ldots, 0) \\ Q_2 &= q(B^1, M_1^1, 0, \ldots, 0, \ldots, 0) \\ Q_3 &= q(B^0, M_1^0, 0, \ldots, 0, \ldots, 0) \\ Q_4 &= q(B^0, M_1^0, \ldots, M_{N-t}^0, 0, \ldots, 0) \end{aligned}$$

By the protocol's correctness, we have that $Q_1 \geq 1 - \mathsf{negl}$. Indeed, the view of honest players in $\mathcal{Q}$ under $B^1, M_1^1, \ldots, M_{N-t}^1, 0, \ldots, 0)$ is indistinguishable from $(B^1, M_1^1, \ldots, M_N^1)$, in which case their output must be 1 since P_N acts honestly. Symmetrically we have $Q_4 \leq \mathsf{negl}$. By Corollary 14, $Q_2 - Q_3 = \mathsf{negl}$. Thus, there is a substantial difference either between Q_1 and Q_2 or Q_3 and Q_4. Without loss of generality we assume the former, i.e., that $|Q_1 - Q_2| \geq 1/3$ (the other cases are similar). By a hybrid argument, we have existence of a $i \in [2, \ldots, N-t]$ such that $|q_0(B^1, M_1^1, \ldots, M_i^1, 0, \ldots, 0) - q_0(B^1, M_1^1, \ldots, M_{i-1}^1, 0, \ldots, 0)| \geq 1/(3(n-t))$. It follows that one of the two q_0 probabilities above must be different by at least $1/(6(n-t))$ from one of the two corresponding q_1 probabilities, e.g., w.l.o.g.:

$$(7) \quad |q_0(B^1, M_1^1, \ldots, M_i^1, 0, \ldots, 0) - q_1(B^1, M_1^1, \ldots, M_{i-1}^1, 0, \ldots, 0)| > \frac{1}{6(n-t)}$$

Back to the main strategy. In round 1, $\mathcal{A}$ sends $(B^1, M_1^1, \ldots, M_i^1, 0, \ldots, 0)$. In the asynchronous phase, after having learned x_1, once $\mathcal{Q}'$ is frozen, and upon unfreezing $\mathcal{Q}$, then: if $x_1 = 0$, $\mathcal{A}$ instructs P_i to play honestly; else if $x_1 = 0$, P_i

is instructed to play as if it did not have received M_i but otherwise honestly. In the first case, the view of honest players follows the distribution which defines the q_0 on the left-hand in (7), while in the latter case, follows the distribution which defines the q_1 on the right-hand in (7). Thus, the joint action of P_i and P_N will has for effect to substantially correlate the second output x_N with x_1. □

References

[Abs+19] M. Abspoel, R. Cramer, I. Damgård, D. Escudero, and C. Yuan. "Efficient Information-Theoretic Secure Multiparty Computation over $\mathbb{Z}/p^k\mathbb{Z}$ via Galois Rings". In: *TCC*. 2019.

[Alb+21] M. Albrecht et al. "Homomorphic Encryption Standard". In: *Protecting Privacy through Homomorphic Encryption*. 2021.

[Ana+18] P. Ananth, A. R. Choudhuri, A. Goel, and A. Jain. "Round-optimal secure multiparty computation with honest majority". In: *CRYPTO*. 2018.

[Ash+12] G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs. "Multiparty Computation with Low Communication, Computation and Interaction via Threshold FHE". In: *EUROCRYPT*. 2012.

[Bar+18] A. Barak, M. Hirt, L. Koskas, and Y. Lindell. "An end-to-end system for large scale p2p mpc-as-a-service and low-bandwidth mpc for weak participants". In: *CCS*. 2018.

[BHN10] J. B. N. Zuzana Beerliova-Trubiniova Martin Hirt. *Almost-Asynchronous MPC with Faulty Minority*. PODC'10. We refer to eprint version 2008/416. 2010.

[BHP17] Z. Brakerski, S. Halevi, and A. Polychroniadou. "Four round secure computation without setup". In: *TCC*. 2017.

[BJMS20] S. Badrinarayanan, A. Jain, N. Manohar, and A. Sahai. "Secure MPC: Laziness Leads to GOD". In: *ASIACRYPT*. 2020.

[BLL20] E. Blum, C.-D. Liu-Zhang, and J. Loss. "Always have a backup plan: fully secure synchronous MPC with asynchronous fallback". In: *CRYPTO*. 2020.

[Bon+18] D. Boneh, R. Gennaro, S. Goldfeder, A. Jain, S. Kim, P. Rasmussen, and A. Sahai. "Threshold Cryptosystems from Threshold Fully Homomorphic Encryption". In: *CRYPTO*. 2018.

[CDKS19] H. Chen, W. Dai, M. Kim, and Y. Song. "Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference". In: *CCS*. 2019.

[CDPW02] R. Canetti, Y. Dodis, R. Pass, and S. Walfish. "Universally Composable Security with Global Setup". In: *TCC*. 2007.

[CGHZ16] S. Coretti, J. A. Garay, M. Hirt, and V. Zikas. "Constant-Round Asynchronous Multi-Party Computation Based on One-Way Functions". In: *ASIACRYPT*. 2016.

[CH18] H. Chen and K. Han. "Homomorphic Lower Digits Removal and Improved FHE Bootstrapping". In: *EUROCRYPT*. 2018.

[Chi+16] I. Chillotti, N. Gama, M. Georgieva, and M. Izabachene. "Faster fully homomorphic encryption: Bootstrapping in less than 0.1 seconds". In: *ASIACRYPT*. 2016.

[Cho+13] A. Choudhury, J. Loftus, E. Orsini, A. Patra, and N. P. Smart. "Between a Rock and a Hard Place: Interpolating between MPC and FHE". In: *ASIACRYPT*. 2013.

[Cho+17] A. R. Choudhuri, M. Green, A. Jain, G. Kaptchuk, and I. Miers. "Fairness in an Unfair World: Fair Multiparty Computation from Public Bulletin Boards". In: *CCS*. 2017.

[Dam+21] I. Damgård, B. Magri, D. Ravi, L. Siniscalchi, and S. Yakoubov. "Broadcast-Optimal Two Round MPC with an Honest Majority". In: *CRYPTO*. 2021.

[DHL21] G. Deligios, M. Hirt, and C.-D. Liu-Zhang. "Round-efficient byzantine agreement and multiparty computation with asynchronous fallback". In: *TCC*. 2021.

[FS01] P.-A. Fouque and J. Stern. "One Round Threshold Discrete-Log Key Generation without Private Channels". In: *PKC*. 2001.

[FV12] J. Fan and F. Vercauteren. "Somewhat practical fully homomorphic encryption." In: *IACR ePrint* (2012).

[GHS12] C. Gentry, S. Halevi, and N. P. Smart. "Homomorphic evaluation of the AES circuit". In: *CRYPTO*. 2012.

[GIKR02] R. Gennaro, Y. Ishai, E. Kushilevitz, and T. Rabin. "On 2-Round Secure Multiparty Computation". In: *CRYPTO*. 2002.

[GLS15] S. Dov Gordon, F.-H. Liu, and E. Shi. "Constant-Round MPC with Fairness and Guarantee of Output Delivery". In: *CRYPTO*. 2015.

[GMP19] N. Genise, D. Micciancio, and Y. Polyakov. "Building an Efficient Lattice Gadget Toolkit: Subgaussian Sampling and More". In: *EUROCRYPT*. 2019.

[GMPS21] V. Goyal, E. Masserova, B. Parno, and Y. Song. "Blockchains Enable Non-Interactive MPC". In: *TCC*. 2021.

[GMW91] O. Goldreich, S. Micali, and A. Wigderson. "Proofs That Yield Nothing but Their Validity or All Languages in NP Have Zero-Knowledge Proof Systems". In: *J. ACM* (1991).

[GO14] J. Groth and R. Ostrovsky. "Cryptography in the multi-string model". In: *Journal of cryptology* 27.3 (2014), pp. 506–543.

[GPS19] Y. Guo, R. Pass, and E. Shi. "Synchronous, with a Chance of Partition Tolerance". In: *CRYPTO*. 2019.

[GSW13] C. Gentry, A. Sahai, and B. Waters. "Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based". In: *CRYPTO*. 2013.

[HNP05] M. Hirt, J. B. Nielsen, and B. Przydatek. "Cryptographic Asynchronous Multi-party Computation with Optimal Resilience". In: *EUROCRYPT*. 2005.

[Kim+20] E. Kim, J. Jeong, H. Yoon, Y. Kim, J. Cho, and J. H. Cheon. "How to Securely Collaborate on Data: Decentralized Threshold HE and Secure Key Update". In: *IEEE Access* (2020).

[Liu+20] C.-D. Liu-Zhang, J. Loss, U. Maurer, T. Moran, and D. Tschudi. "MPC with Synchronous Security and Asynchronous Responsiveness". In: *ASIACRYPT*. 2020.

[LPR13a] V. Lyubashevsky, C. Peikert, and O. Regev. "On Ideal Lattices and Learning with Errors over Rings". In: *J. ACM* (2013).

[MTBH20] C. Mouchet, J. Troncoso-Pastoriza, J.-P. Bossuat, and J.-P. Hubaux. "Multiparty homomorphic encryption from ring-learning-with-errors". In: *PoPETS* (2021).

[MW16] P. Mukherjee and D. Wichs. "Two round multiparty computation via multi-key FHE". In: *EUROCRYPT*. 2016.

[Par21] J. Park. *Homomorphic Encryption for Multiple Users with Less Communications*. IACR ePrint 2021/1085. 2021.

[PR18] A. Patra and D. Ravi. "On the Power of Hybrid Networks in Multi-Party Computation". In: *IEEE Transactions on Information Theory* (2018).

[RSY21] L. Reyzin, A. D. Smith, and S. Yakoubov. "Turning HATE into LOVE: Compact Homomorphic Ad Hoc Threshold Encryption for Scalable MPC". In: *CSCML*. 2021.

A Further Details on Related Works

A.1 Related Approaches with FHE

A.1.1 Details on the Approach of [GLS15] (-1) players receive a uniform random string. (0) based on the received string, players generate and publish GSW public keys. (1) each player P_i encrypts its input under an encryption scheme specific to it, which we may denote Flexible_i, and which takes as parameter the published keys, then broadcasts it. (1) *In parallel*, players perform the second event of a DKG, establishing a common (N,t)-TFHE public key. (2) Upon learning this key, players are able to Transform, locally and deterministically, the Flexible ciphertexts into TFHE ciphertexts under the common key. Then players proceed with local evaluation of the circuit; then finally threshold decryption, which can be Responsive, by our observation.

More precisely, Flexible_i of player i goes as follows: on input m_i, output the vector $\hat{ct}_i = (ct_{i,1}, ct_{i,2}, ..., ct_{i,N})$, where $ct_{i,i}$ is a ciphertext of input m_i under P_i's GSW public key, while the other cti, js are GSW ciphertexts of 0 under each P_j's public key. By construction, since every Flexible_i scheme has its secret key known by the corresponding P_i, it has threshold 0 with respect to the players, which prevents it from being used by external input owners. By contrast, our Ad-Hoc scheme is unique and furthermore (N,t)-threshold, which enables usability by external input owners. Another particularity of Flexible ciphertexts is that all the N GSW ciphertexts contained are actually generated with the same secret randomness. Thus, given that half of these ciphertexts are encrypted under GSW public keys which were generated by the adversary $\mathcal{A}$, they a priori give $\mathcal{A}$ an extra advantage to guess the plaintext of $ct_{i,i}$. For the security of GSW to hold, their public keys and randomness are thus scaled larger, in order to apply the leftover-hash-lemma (LHL).

However this technique is not efficiently transposable to our setting, for the following reasons. To start with, referring to §4, suppose that the adversary is given one (or several) BFV encryptions of 0 under semi-maliciously generated key(s) (a, b_i), i.e., $\left(u\, b_i + e_{0,i}^{(Enc)}, u\, a + e_{1,i}^{(Enc)}\right)$, which would all be generated with the same secret randomness $u \leftarrow \mathscr{X}_q$. Then this may provide it with distinguishing advantage when given a BFV encryption of some m under some honest key (a, b_j) which would re-use the same randomness u. One could possibly think of a fix, e.g., adapting BFV by specifying that the first component of the public keys, $a := \mathbf{a}[0]$, would be instead vectors with coordinates in R_q, and encryption randomness $\mathbf{u}$ equal to a random vector with entries in R_q. This new setting would however lessen the complexity gain with respect to GSW. In addition, there is still the problem that, even in this setting, [Dac+21, §1] evidenced a counterexample showing that the LHL does not hold in general (a leakage of $1/n$ of the secret randomness which would make the outcome far from indistinguishable from uniform). They also point that [LPR13b, Cor 7.5] showed that a weaker version of LHL, denoted "regularity", does apply in this setting, nonwithstanding the previous leakage issue, in the case where the distribution of secret

randomness would be Discrete Gaussian with sufficiently large parameter. Concretely, we would apply "regularity" to $\mathbf{A}.\mathbf{u}$, where A would be a matrix with N rows encoding all public keys. The problem is that, for the applicability of "regularity", it is required that $\mathbf{A}$ be sampled uniformly, whereas in our setting we have t keys in $\mathbf{A}$ which are semi-maliciously generated, furthermore possibly depending on the other $t+1$ honest keys. Thus, this situation could potentially leak substantial information on $\mathbf{u}$, and thus potentially enable to distinguish the outcome from random uniform, such as mentioned above [Dac+21, §1].

As a final remark on [GLS15], notice that, on the face of it, their DKG is specified over pairwise channels, thus a player which would abort after sending only part of the messages it is meant to send, would leave honest players with inconsistent views on which contributions to the threshold key should be taken into account. However this is handled by [GLS15, Remark 4.1], who observe that players can be instructed to broadcast the set of their messages, encrypted under each recipient's public key. From this perspective, their DKG, as well as the one of [FS01], can be seen as a particular case of $\mathcal{F}_{\mathrm{DLC}}$ to do a sum of contributions to the secret threshold decryption key, for the particular parameter of a modulus q equal to a prime larger than $N+1$ [GLS15, §B, §C].

A.1.2 Based on Multi-key FHE and on (N, N)-Threshold FHE The recent work [CDKS19] constructs a multikey variant (MFHE) of the BFV [FV12] and CKKS [Che+17; LM21] schemes, with ciphertexts of linear size in the number N of players (down from quadratic [MW16; BHP17]). However the size of the ciphertexts is still of order of magnitude N times greater than a single-key (N, N)-TFHE scheme and computation complexity N^2 times greater. Recently, [Hye22] reduced the overhead in computation complexity to linear in N.

Turning to (N, N)-Threshold FHE, this particular threshold $t := N$ yields a simplification over general (N, t)-TFHE, which is that establishing the threshold key can be done in one non-responsive event instead of 2: each player generates a key pair and publishes the public key, the threshold key is then defined as the sum of public keys. The secret decryption key is equal to the sum of the secret keys, which by definition constitutes an additive secret sharing.

However, both MFHE and (N, N)-TFHE suffer from nonconstant worst case latencies, both total and to responsiveness. Indeed if one player aborts in the threshold decryption, then it fails, by definition of N out of N access structure. *Assuming synchrony*, then one could imagine a compilation of such protocols into ones guaranteeing GOD. Namely: require players to send their decryption shares via terminating reliable broadcast. Then, if some instances of broadcast for some players returned $\perp$, discard them and restart the whole protocol with the remaining players. However, waiting for these time-outs is by definition not responsive. Recall that we have another non-responsive event at the beginning of every execution, which is the time-out after which players which did not broadcast an encryption of their input are discarded from the protocol. Thus, in the case of $N/2$ consecutively aborting players in the distributed decryption, the execution is restart $N/2$ times.

There are moreover other sources of potential slowdown in specific approaches, such as the (N, N)-TFHE of [Par21]. There, generation of the relinearization key is done after the threshold key is known, and completes only if all players who contributed to generation of the threshold key, also participate. Notice that there is the same issue for the distributed generation of the bootstrapping key. Thus, if one had wanted to enable GOD in [Par21], one would need to also require that players send their contribution to the relinearization key via terminating reliable broadcast. Let us observe that this broadcast could be done in parallel of the broadcast of encrypted inputs. Then, if some instances of broadcast for some players returned $\perp$, discard these players and restart the whole protocol with the remaining players.

A.1.3 Based on (N, t)-threshold FHE The work [Ash+12] was the first to introduce (N, t)-TFHE with thresholds t lower than $N - 1$. Unfortunately, their construction adds 2 additional rounds to the protocol as soon as one player aborts. Indeed, in this case it is required that the honest majority reconstructs the player's state and resume the protocol.

The work [Bon+18] also proposed (N, t)-TFHE schemes. However, their §5 leaves unspecified the DKG, while their §6 has the drawback that the encryption and decryption key pair is generated by one entity (typically, one secret owner, before encrypting its secrets), thus it would be unsafe to have other secret owners also encrypt their secret under this same key, which prevents MPC.

A.2 Garbled circuits, and Joint Use with BFV for Numerical Computation

Note that the communication size in [Ana+18] is improved by [Ana+20] to $O(|C| + N^4)$, but with a suboptimal corruption threshold $\frac{1}{2} - \epsilon$, this is why they recommend to use [Ana+18] for small values of N. [Abh21], which provides security with abort, allows external inputs.

Garbled circuits and TFHE benefit from being used together, e.g., in use-cases of machine learning. Indeed, each layer of a neural network has a large circuit of linear gates over arithmetic values, where homomorphic encryption or secret-sharing techniques are very fast, whereas conversion of these linear operations to a boolean circuit is prohibitively expensive. Thus, the typical approach is hybrid: on the one hand, computations of linear functions (FC/CNNs) use either homomorphic encryption (Gazelle [JVC18]) or secret-sharing (MiniONN [LJLA17]). The work [Che+20] somehow combines the two, since HE is used to pre-compute secret-shared authenticated multiplicative Beaver triples of matrices. Among other, they leverage the possibility to encode several plaintexts in a single BFV ciphertext, which will undergo the same homomorphic operations in parallel. This technique is known as "packing" [SV14; KÖ22].

Then, other non-linear functions are computed with garbled circuits.

A.3 Other synchronous / asynchronous hybrid models

The question of minimizing the number of synchronous rounds before an execution of MPC was initiated in [HNP05]. The broadcast of [FN09] uses a few synchronous rounds, then guarantees responsive eventual output delivery, under a honest majority. In information-theoretic MPC with $t < N/3$, [PR18] reached an optimal 3 pre-Responsive synchronous rounds. In the setting of MPC with non-constant latency, then [BHN10] exhibited a protocol with only one all-to-all pre-reponsive broadcasts of encrypted inputs, motivating their terminology "Almost Asynchronous". However, their encryption requires a (N, t)-threshold additively homomorphic encryption scheme: establishing it with a DKG would thus add 2 other pre-Responsive events. [RU21] showed how to bring the total from 3 to 2, by building an ad-Hod threshold additive encryption scheme.

The protocol [BLL20] proceeds by intervals of fixed duration, denoted rounds, of which the number grows with the depth of the circuit. In particular it is not responsive. Since their model is free of primitives such as Byzantine Agreement under honest majority, thus, if the network is asynchronous, they cannot guarantee input provision, nor guaranteed delivery of a correct output, which can possibly be a default value $\perp$.

The protocols [CGHZ16; Coh16] are purely asynchronous, i.e., responsive, thus do not withstand more than $t < N/3$ corruptions. Even if the latter has a trusted setup, withstanding more than $t < N/3$ corruptions is impossible since the protocol is purely responsive.

B Model: Further Formalism and Discussion

B.1 Latency, Ordered Events in Asynchronous Executions, and Responsiveness

We did not further define Responsiveness than just in the Introduction, because it is not part of our Theorems. Instead, Theorem 1 specifies explicitely the consecutive events of our implementation, i.e., 2 consecutive broadcasts, followed by 2 consecutive point to point asynchronous communications. Let us notice that in the Introduction we allowed the use of secure message transmitting in responsive executions, although in our protocol we use only authenticated message transmitting.

Let us now further define (Latency to) Responsiveness.

B.1.1 Latency, Ordered Events in Asynchronous Executions, adapted from Lamport in the Malicious Setting Let us recall the measurement of latency in an asynchronous execution, following Lamport's mainstream definition [Lam78], e.g., as exposed in [Lam06]. We slightly adapt it to our model with malicious corruptions and ideal functionalities. The reader may as well skip it and consider instead the notion of "asynchronous rounds" in [CGHZ16]. We say than an event b is *Consecutive to event* a, also denoted *a happened before*

b if there is a chain of causality of events originating from a and leading to b, e.g., the sending of a message at a which is delivered at b. Two events are in *Parallel* if none is consecutive to the other. Consider a chain of consecutive events. Its length is defined as the number of consecutive *sub*chains with starting point: {sending or broadcasting of a message by a honest player}, followed by {arbitrary consecutive events}, and whose ending point is: {output from some functionality to a honest player}, e.g., output from a broadcast. The longest chain of consecutive events before some event e is denoted as the "depth" before e, which we dub as the *Latency to* e.

Definition 15 (Latency to Responsiveness). *in an execution, is the maximum, over all honest $S \in \mathscr{P} \cup \mathscr{L}$, of the depths of the following events e_S. e_S is the event where S, from this point, does not anymore: access other functionalities than $\mathcal{F}_{AUTH}$, and does not follow instructions to wait for a given time delay, be it measured by a local or global clock, or a UC glock [CGHZ16], and does not compute proof of works [GKOPZ20].*

Notice that in our protocol $\mathcal{F}_{\mathrm{DLC}}$, the command LCOpen is implemented via $\mathcal{F}_{\mathrm{AUTH}}$ only, thus, when implementing $\Pi_C^{\mathcal{F}_{\mathrm{DLC}}}$ with Π_{DLC}, the command LCOpen is responsive.

B.1.2 Related Notion: Optimistic Responsiveness The terminology Responsiveness was coined in [PS18], as the guarantee of a latency independent from an a priori upper-bound on the network delay. However, since purely responsive protocols are limited to $t < N/3$, they consider instead protocols in which responsiveness is achieved only in optimistic cases, which is a related guarantee that they denote Optimistic Responsiveness. In particular, players in their protocol are instructed to wait, roughly, to hear from $3/4$ of players in some committee, until a time-out, after which they switch to a synchronous fallback protocol. But $3/4$ is strictly more than the expected proportion of corrupt players in the committee, which is $t < N/2$. Thus, these instructions do not satisfy Responsiveness, i.e., do not allow players to proceed as soon as they receive $N-t$ honest-looking messages.

B.1.3 A Conflicting Notion of Responsiveness In [Cam+16], the problem of the adversarial scheduling of activates of various functionalities in the UC model is discussed. They propose a model in which specific methods of some functionalities have a specially high priority, in the sense that the adversary, upon receiving an activation request for them, is forced to activate them before it can activate any normal priority method of any functionality. They denote adversaries and/or Environments respecting these rules as "Responsive". This notion is thus unrelated to our terminology. By contrast, in our model we do not assume any priority methods nor responsive Environment. The purpose behind [Cam+16] is to remove artefacts of the UC model that give the adversary a power to block some events, although there is no counterpart of this power

in known implementations. An example which concerns our protocol (§5.6) is the UC formalism of NIZK in [GOS06], also recalled in §B.9: $\mathcal{F}_{NIZK}$ waits to receive from the adversary the string that constitutes the NIZK. Although in any actual implementation, a honest player which would like to output a NIZK of some witness which it owns, can do it without waiting. This artefact could potentially provoke an artificial safety issue, in synchronous protocols like [Ash+12] in which, when a player is not heard of in time, then other players may open its state. Therefore, in [Ash+12] it may seem reasonable to cast UC NIZK in the model of [Cam+16], as having high priority.

This is fortunately not an issue in our model. Indeed, $\mathcal{F}_{NIZK}$ shows up in our compiler to malicious security as follows. Every player and owner is instructed to append NIZKs to its messages in (1),(2) and (3). Thus, if the adversary blocks a player from receiving its NIZK from $\mathcal{F}_{NIZK}$, this has exactly the same effect as if the adversary did release the NIZK but did not activate delivery of any message sent by the player. In turn, this scenario is already captured by our UC security proof.

B.2 More on Terminating Reliable Broadcast BC with Possibly External Senders

B.2.1 Comments on Implementations Implementations of $\mathcal{F}_{\mathrm{BC}}^{P,\mathscr{P}}$ for $P \in \mathscr{P}$ are mainstream since [LSP82]. They necessarily assume that P and $\mathscr{P}$ are able to start the protocol synchronously, and that P is provided with authenticated channels to $\mathscr{P}$ guaranteeing fixed public delivery delay δ. Otherwise, one would need to relax the specification into allowing that the output be not always the one of S whenever honest, as the case in weak/partial synchronous models [GPS19; ANRX21]. Indeed, a S which lags behind could not be told apart from a dishonest S which remains silent. In our generalized context with possibly external receivers, implementation of bPKI^P can be obtained directly from $\mathcal{F}_{\mathrm{BC}}^{P,\mathscr{P}}$, as: players run $\mathcal{F}_{\mathrm{BC}}^{P,\mathscr{P}}$, then, upon receiving their output, each player sends it to all external receivers, i.e., in $\mathscr{L}$. Each $\ell \in \mathscr{L}$ waits to receive $t+1$ times the same value, then outputs it. In our generalized context with a possibly external sender $\ell \in \mathscr{L}$: an implementation of BC^ℓ can be obtained assuming (i) channels from ℓ to $\mathscr{P}$ with guaranteed delay δ, and that (ii) ℓ and $\mathscr{P}$ are able to start synchronously, by: ℓ sends its input to all $\mathscr{P}$, then after δ, $\mathscr{P}$ perfom Byzantine consensus with inputs what they received from ℓ ($\perp$ if none). Notice that our protocol will instruct all players and owners to act as senders in BC upon completion of all instances of bPKI. As discussed above, implementation of BC thus necessarily requires senders and $\mathscr{P}$ to start synchronously at a point in time after completion of these instances.

B.2.2 Sub-sessions For sake of UC analysis, in our MPC protocol we specify multiple broadcast instances per sender. This is why we formalize $\mathcal{F}_{\mathrm{BC}}^{S,\mathcal{R}}$ with sub-session identifiers denoted ssid. Importantly, we force $\mathcal{F}_{\mathrm{BC}}^{S,\mathcal{R}}$ into allowing at most one ssid per sender, to prevent two players from receiving different outputs from

a corrupt sender for the same ssid. In practice in our protocol the ssid is the label of the variable which is broadcast. Furthermore, we make the abuse of notation, in our protocol, to have one sender concatenate multiple broadcasts instances of several variables at once, likewise for the input command of $\mathcal{F}_{\text{DLC}}$. Indeed this is how the protocol would be efficiently implemented. In a model allowing that, an input owner can possibly be logically identified to some player, and thus both would either be simultaneously honest or corrupt. One could furthermore have them concatenate all their broadcasts in (1).

B.3 More on our bPKI, and the { Bulletin board / Untrusted / bare } PKI model

B.3.1 Comparison with Canetti's certification authority Recall the we formalized bPKI as a mere UC Terminating reliable broadcast, with possibly external receivers. The closest UC definition of bPKI known to us is the "certification authority" $\mathcal{F}_{CA}$ of Canetti [Can04]. There, it is presented in the same use-case, namely, a service enabling players to record their public keys. In appearance, its formalization is different than ours since it provides delayed output only to players who requested it. But it actually seems to implement our bPKI, by having players repeatedly query an output. Notice the conflict of terminology in [Can04], in which the symbol $\perp$ means that the instance did not terminate yet, instead of, in our definition, did terminate with adversarially chosen output $\perp$.

B.3.2 PKI assumptions in the literature The bPKI functionality, for our usage limited to publication of public keys, could be traded by the assumption denoted {Bare/Untrusted/Bulletin board} public key setup (PKI)" [Ana+18; BCG21; GJPR21]. The PKI model was first sketched in [CGGM00, §6], then denoted as "bare PKI" in [Ana+18]. It is renamed as "untrusted PKI" in [GJPR21]. As specified in [CGGM00, §6.1][GPS19; Dam+21], the PKI model is slightly stronger than ours, since it abstracts-out all the implementation constraints discussed in §B.2, in a way which could be phrased as: (i) all instances of bPKI are assumed to terminate before a public time denoted $t = 0$, (ii) players (and owners) are then able to reset their clocks synchronously at $t = 0$, which, e.g., enables them to subsequently run implementations of BC. By contrast, the implementation of $\mathcal{F}_{\text{BC}}$ in [FN09] does not guarantee delivery within a fixed delay (the "$t = 0$"), only eventually.

Notice that, formalized like this, the "$t = 0$" is related to the notion of "synchronization point" in [Dam+09; Liu+20].

Likewise, in the specific case of protocols for Byzantine agreement, [BCG21] also assume access to the board only before players are assigned their inputs, which is the same assumption in our case. We refer to [BCG21] for a comparison of bulletin-board PKI with more demanding setup assumptions. Notice that our generalization, where inputs are formally assigned to owners instead of players, parallels the regime of state machine replication in which commands originate from external lightweight "clients".

B.3.3 The power of { Bulletin board / Untrusted / bare } PKI in MPC By construction, bPKI does not perform any check on the written strings, it displays them to all players. Thus it is strictly weaker than the setup denoted as "registered PKI", or KRK, in [CDPW02], where it is proven to have strictly more power. Notice that implementing KRK without $\overline{\mathcal{G}}_{\text{URS}}$ would, in turn, require an extra event before bPKI in which players would publish multi-string CRS. Fortunately KRK is not required in our protocol, only plaintexts are extracted in our UC proof, not secret keys.

In [GJPR21] it is proven that MPC in two rounds is impossible in the plain model, even with identifiable abort. However, it is feasible assuming the bare/untrusted PKI Model. Recall that in our protocol, we instead implemented the bare PKI with the non-Responsive event, denoted $\textcircled{0}$, consisting in N terminating instances of bPKI in parallel.

To the best of our knowledge, bPKI is the minimal setup necessary in all implementations of synchronous broadcast for $t \geq N/3$, since the seminal [LSP82]. Also, [Bor96, Thm 1] shows that the relaxation of bPKI denoted as "local setup", precludes synchronous broadcast for $t \geq N/3$. [Local setup means that a corrupt player can possibly make bPKI display different strings to different players. However, it cannot claim for itself a string previously published by a honest player.]

To be complete, let us mention that Borcherding's impossibility is circumvented with the additional assumption that the majority of a restricted resource, e.g. the computing power (or, alternatively, storage space) is in the hands of honest players, which we may denote as the "proof of work (PoW) model". There, both [Daz+08; Agg16] implement what is denoted as a "pseudonymous" PKI, i.e., a mechanism that outputs to all honest players a single set of public keys, with possibly several keys assigned to the same players, while guaranteeing that the majority of keys were issued by, and thus are owned by, honest players. In the same PoW model, [GKOPZ20] implement an actual untrusted PKI, i.e., the $\mathcal{F}_{CA}$ of [Can04], which they denote $\mathcal{F}_{REG}$.

B.3.4 A Related Notion of Bulletin Board The terminology "bulletin board" is introduced in [Cho+17], where it is used as a support of communication for MPC, as well as in [GMPS21]. Therefore, this usage contrasts with the mainstream usage of a "bulletin board PKI", which is also ours, in which players cannot write on the board after the time $t = 0$ when owners receive their inputs. However, the formalization of [Cho+17; GMPS21] seems identical to our bPKI. The only formal difference is that they specify synchrony, with possibly offline players. Thus, in their terminology, our "eventual delivery" is replaced by something which could be phrase as "provides output even to players which were offline while the bulletin-board was written on".

B.4 $\mathcal{F}_{\text{AUTH}}$

In the two last (asynchronous) steps of our MPC protocol, when $\mathcal{F}_{\text{DLC}}$ is instantiated with Π_{DLC}, each player is instructed to send the same opening share

$\mathcal{F}_{\text{AUTH}}^{S,R}$

On input $(\mathsf{input}, \mathsf{ssid}, v)$ from S: then provide R with delayed output (ssid, v).

Fig. 5: (Asynchronous, Public) Authenticated message transmitting for S and R.

to all using $\mathcal{F}_{\text{AUTH}}$. Notice that nothing prevents corrupt players from sending different opening shares to different players. However, when sending a share, semi-maliciously corrupt players must exhibit an input tape containing a pair of: secret key and a key noise, compatible with the $\mathbf{b_i}$ which they broadcast in (1). Thus, for whatever compatible pair which they could exhibit, the decryption share coming with it is necessarily correct.

Notice that, in our protocol, players are instructed to publish their public key on bPKI at the beginning, thus $\mathcal{F}_{\text{AUTH}}$ could actually have been implemented from non-authenticated channels.

Notice that, contrary to $\mathcal{F}_{\text{BC}}$ and bPKI, there is no $\mathsf{activate}$ command for the output from a corrupt sender in $\mathcal{F}_{\text{AUTH}}$. Hence, our definition of "complete execution" does not require termination of $\mathcal{F}_{\text{AUTH}}$ instances for corrupt senders. Requiring so would have lead to a definition of "guaranteed output delivery" which would not even have withstanded fail-stop adversaries.

On the other hand, in order to guarantee GOD in our MPC protocol, we need implementations of $\mathcal{F}_{\text{AUTH}}$ with guaranteed eventual delivery from a honest sender. Notice that without this guarantee, then no asynchronous protocol could have guaranteed termination.

B.5 $\overline{\mathcal{G}}_{\text{URS}}$

$\overline{\mathcal{G}}_{\text{URS}}$ is a particular case of $\mathcal{F}_{crs}$ in [CLOS02].

$\overline{\mathcal{G}}_{\text{URS}}^{\kappa}$

On input query from all honest players in $\mathscr{P}$, then samples a sequence of κ bits uniformly at random then outputs it to each player $P \in \mathscr{P}$, then halts.

Fig. 6: Uniform Random String.

On the one hand, MPC under honest majority enables to UC implement fair coin tossing. Thus $\overline{\mathcal{G}}_{\text{URS}}$ could have been implemented, at the cost of extra preliminary non-responsive steps. Notice that optimized implementations exist, but at the cost of more assumptions. E.g., [CD20] requires a CRS for generation of NIZKs, while [Kia+21] requires an initial seed.

On the other hand, when upgrading $\overline{\mathcal{G}}_{\text{URS}}$ to a global setup, then it is not proven in general if one can safely implement a global setup by using any protocol proven UC secure, e.g., see [BHZ21].

B.6 More on Guaranteed Eventual Output Delivery

A more complex UC formalization of guaranteed eventual output delivery in an asynchronous network was done by [CGHZ16], then in [Liu+20].

B.7 Straightforward Generalizations of $\mathcal{F}_C$

To start with in our protocol, messages of input owners do not depend on the actual circuit to be computed. Thus, our protocol actually achieves the "delayed function" property ([Ana+18]), i.e., it implements a more general functionality to which honest players can possibly provide the circuit to be computed *after* the broadcast instances from all owners terminated, possibly with some $\perp$ or badly formed messages for some of them. We do formalize and implement such a functionality in §3 for the specific case of linear forms, because we need this possibility in the implementation of MPC protocol.

As such, our definition imposes that every honest player, and only them, outputs. However, it straightforwardly generalizes to protocols implementing functionalities which deliver outputs to arbitrary receivers, be them included, overlapping, or disjunct, from the set $\mathscr{P}$ of players. Interestingly, different feasibility bounds hold for general receivers, e.g., the case of Solitary MPC [Bad+21].

B.8 Reminder of the UC model, and our Simulators

Consider a protocol Π, a functionality $\mathcal{F}$ and any PPT environment $\mathscr{E}$ which can interact with either one or the other of the following protocols, without being informed which of them. In every execution, $\mathscr{E}$ may provide inputs to honest players, may provide instructions to the adversary, and may observe the outputs of players. At some point of every execution, $\mathscr{E}$ must output a bit. In the first, denoted "real" $REAL_{\Pi}$, $\mathscr{E}$ interacts with the dummy adversary $\mathcal{A}$, and honest players follow the actual protocol Π. In the second, denoted "ideal" $IDEAL_{\mathcal{F},\mathcal{S},\mathscr{E}}$, $\mathscr{E}$ interacts with an adversary $\mathcal{S}$, while honest players are connected only to $\mathcal{F}$. Following [Can01], we say that protocol Π UC emulates $\mathcal{F}$ if there exists a PPT machine, denoted as the simulator $\mathcal{S}$, such that for any such $\mathscr{E}$, the gap of probabilities of outputing 1 when faced with an execution of the first protocol, and when faced with an execution of the second, is negligible. Notice that this definition, with only the dummy adversary, is easily seen, and proven in [Can01, §4.3.1], to be equivalent to UC emulation against any adversary.

B.8.1 Simulators in our UC Proofs Our simulators $\mathcal{S}$, unless specified, have the following high level behavior. Initiate in its head: a set of N players and $|\mathscr{L}|$ Owners, along with the ideal functionalities with which they are meant to interact in the actual protocol, excepted $\overline{\mathcal{G}}_{\text{URS}}$, since it is an external resource. Upon corruption requests from $\mathscr{E}$, $\mathcal{S}$ labels in turn the corresponding simulated players or owners as corrupt. Upon every output from a simulated functionality to a simulated corrupt player, or, upon every network request from a simulated functionality to $\mathcal{S}$, then $\mathcal{S}$ immediately updates $\mathscr{E}$ with it, at would have done the actual dummy adversary.

We will abuse notations such as: "honest P sends message to corrupt Q", where we actually mean that, in the simulated execution, $\mathcal{F}_{\text{AUTH}}^{P,Q}$ requests (network,m)

to $\mathcal{S}$, then, upon receiving an instruction to activate from $\mathscr{E}$, then the simulated $\mathcal{F}_{\text{AUTH}}^{P,Q}$ outputs to simulated corrupt Q, then immediately updates $\mathscr{E}$ with this reception by Q, as the dummy adversary would have done.

The same abuses of notation apply to other interactions of simulated corrupt players with other simulated functionalities, such as $\mathcal{F}_{\text{DLC}}$ in our proof of Th 10. In particular, when we say that "$\mathcal{F}_{\text{DLC}}$ delivers some delayed output c to some corrupt player Q", then we actually mean that: $\mathcal{S}$ creates a value c; simulates towards $\mathscr{E}$ that it received a request (a "OutValReq") from $\mathcal{F}_{\text{DLC}}$ (which does not exist) for delayed output of c. Then, upon receiving an instruction to activate from the Environment, then $\mathcal{S}$ simulates towards $\mathscr{E}$ that it was notified by Q (which does not exist) its reception of c from $\mathcal{F}_{\text{DLC}}$.

B.9 UC Non-Interactive Zero-knowledge (NIZK) functionality

Non-Interactive Zero-knowledge $\mathcal{F}_{NIZK}$, following exactly [GOS06], upon request of a prover P exhibiting knowledge of some witness w verifying a public statement $(x, *) \in R$, delivers to P the *delayed output* of a string π which can subsequently be checked by any verifier to verify that P does know some w verifying $(x, w) \in R$.

$\mathcal{F}_{NIZK}$

The functionality is parameterized with an NP relation R of an NP language L, and a prover P.

Proof: On input $(prove, \mathsf{sid}, \mathsf{ssid}, x, w)$ from P, ignore if $(x, w) \notin R$. Send $(\mathsf{network}, proof, x)$ to $\mathcal{A}$ and wait for answer $(\mathsf{activate}, \pi)$. Upon receiving the answer store (x, π) and send $(proof, \mathsf{sid}, \mathsf{ssid}, \pi)$ to P.

Verification: On input $(verify, sid, ssid, x, \pi)$ from $V \in \mathscr{P}$, check whether (x, π) is stored. If not send $(\mathsf{network}, verify, x, \pi)$ to and wait for an answer $(\mathsf{activate}, witness, w)$. Upon receiving of the answer, check whether $(x, w) \in R$ and in that case, store (x, π). If (x, π) is stored, return $(\mathsf{verification}, \mathsf{sid}, \mathsf{ssid}, 1)$ to V, else return $(\mathsf{verification}, \mathsf{sid}, \mathsf{ssid}, 0)$.

Fig. 7: Non-Interactive Zero-knowledge functionality

UC implementations of $\mathcal{F}_{NIZK}$ exist, which do not require honest majority [DDOPS01], but at the cost of requiring a uniform random string (URS). Notwithstanding that [DDOPS01] allow the same URS to be reused in concurrent executions, the bottom-line is that the URS needs to be part of a local setup in their implementation. Without a honest majority assumption, then [CDPW02] prove that UC NIZK are non-implementable in the global common random string model, i.e., which we formalized as $\overline{\mathcal{G}}_{\text{URS}}$ in the particular case where the string is uniform.

Fortunately, the need of a URS can be escaped under honest majority, provided access to bPKI, thanks to the technique denoted multi-string CRS [GO14; BJMS20].

C More on $\mathcal{F}_{\mathbf{DLC}}$ and Secret Sharing over Rings

Notice that the specification of $\mathcal{F}_{\mathrm{DLC}}$ is related to the one denoted F_{com} in [CDN15, p. V].

The minor difference is that we merged in one single command LCOpen the three separated commands of F_{com} which were Addition, scalar Multiplication and Open.

The major difference is that $\mathcal{F}_{\mathrm{DLC}}.\mathsf{LCOpen}$ produces an output as soon as it receives request from any $t+1$ players, i.e., is responsive. By contrast, F_{com} needs request from all $t+1$ honest players to open a value.

C.1 Detailed construction of the linear secret sharing over R_q

Now for the details: consider an irreducible polynomial $\overline{Q(T)} \in \mathbb{F}_p[T]$ of degree $d := \lceil \log(N+1) \rceil$, then an arbitrary lift Q in $\mathbb{Z}/q\mathbb{Z}$. Embed R_q in the R_q-algebra $S := R_q[T]/Q$, which we may also denote as $\mathrm{Gal}(R_q, d)$. Now, in $S = \mathrm{Gal}(R_q, d)$ we have the sub-ring $B := \mathbb{Z}/q\mathbb{Z}[T]/Q$, denoted $\mathrm{Gal}(\mathbb{Z}/q\mathbb{Z}, d)$ the "Galois ring extension of degree d of $\mathbb{Z}/q\mathbb{Z}$" [Abs+19]. [If it is not clear yet that B is a sub-ring: apply [AM69, ex 6 p32] to $A := \mathbb{Z}/q\mathbb{Z}$, $M := R_q$, $B = \mathrm{Gal}(\mathbb{Z}/q\mathbb{Z}, d)$]. But, recall from [Abs+19] that $\mathrm{Gal}(T/qT, d)$ has the desired property to contain at least $N+1 = 2^{\log(N+1)}$ elements, denoted $(\alpha_0 := 0, \alpha_1, ..., \alpha_N)$ such that all pair-wise differences $\alpha_i - \alpha_j$ for $i \neq j$ are invertible [Concretely: choose the $(\alpha_i)_{i=0,...,N}$ as arbitrary lifts modulo q of distinct elements of the finite field $\mathbb{F}_{p^d}$]. Thus we can apply to S and these evaluation points $(\alpha_i)_{i=0,...,N}$ the same previous construction as for LS_{R_q}. We obtain a S-linear secret sharing scheme over S: LS_S, in particular by the ring inclusion $R_q \hookrightarrow S$, a R_q-linear secret sharing over R_q. Each share, which is in S, can be encoded as d elements of R_q. [Concretely, $LS_S(s)$ is defined as: sample P at random in $S[Y]_t^{(s)}$, then output $[P(\alpha_i)]_{i\in[N]}$. Then for reconstruction use the Lagrange polynomials $\Pi_{j\neq i}(X - \alpha_j)/(\alpha_i - \alpha_j)$.]

C.2 Detailed Proofs

Recall that, although in general each share is in R_q^d, where $d := \lceil \log_p(N+1) \rceil$ when q is some power of p, for simplicity, in the remaining we do as if shares were in R_q.

C.2.1 Uniformity of any t shares of any given secret.

Property 16. *For every $s \in R_q$, for any subset of t indices $\mathcal{I} \subset [N]$, the distribution of shares $(s^{(i)})_{i\in\mathcal{I}}$ output by $\mathsf{LS}_{R_q}.\mathsf{Share}(s)$ is $U(R_q^t)$.*

Proof. By surjectivity (isomorphism) of $\mathsf{Eval}_{\{0\}\cup\mathcal{I}} : R_q[Y]_t \to R_q^{t+1}$ for any t-sized $\mathcal{I}$, we have surjectivity (isomorphism) of $\mathsf{Eval}_{\mathcal{I}} : R_q[Y]_t^{(s)} \to R_q^t$ for any fixed $s \in R_q$. Furthermore, the map $\mathsf{Eval}_{\mathcal{I}}$ being also linear, we have in conclusion that it maps the uniform distribution onto the uniform distribution. □

C.3 Proof of IND-CPA of Public Secret Sharing

We are going to show a bit more than IND-CPA. We consider a game in which the adversary $\mathcal{A}_{\mathsf{PSS}}$ can select t indices $\mathcal{I} \subset [N]$ to corrupt, without any further information given to it at this stage. Then the oracle $\mathscr{O}_{\mathsf{PSS}}$ generates $[N]$ public keys for PKE and shows them to $\mathcal{A}_{\mathsf{PSS}}$. Furthermore, it reveals to $\mathcal{A}_{\mathsf{PSS}}$ the t secret keys with indices in $\mathcal{I}$. Then $\mathscr{O}_{\mathsf{PSS}}$ tosses a bit b and subsequently responds to encryption requests from $\mathcal{A}_{\mathsf{PSS}}$ as follows. Either $(b = 1)$ then $\mathscr{O}_{\mathsf{PSS}}$ returns to $\mathcal{A}_{\mathsf{PSS}}$, upon each plaintext s chosen by $\mathcal{A}_{\mathsf{PSS}}$, a correctly generated PSS of s of its choice, or, if $(b = 0)$, a sample of the following distribution, which we denote $\mathcal{V}$ such that:

- the entries in $\mathcal{I}$ are PKE encryptions of uniform independent values in R_q;
- the remaining entries are PKE encryptions of 0.

Since this distribution $\mathcal{V}$ is independent from s, the indistinguishability that we claim indeed implies IND-CPA.

We are going to bound the advantage by any adversary $\mathcal{A}_{\mathsf{PSS}}$, by the maximum advantage of an adversary $\mathcal{A}_E$ against oracle $\mathscr{O}_E$ of the following $(N-t)$-keys variant indistinguishability game for PKE. The latter is upper-bounded by $(N-t)$ times the advantage for one-message indistinguishability, see e.g. [BS20, Thm 5.1] or our proof of Corollary 7, which is identical. $\mathscr{O}_E$ samples $(N-t)$ PKE public keys $(\mathsf{pk}_h^{\mathsf{PKE}})_{h\in\mathcal{H}}$ which it gives to $\mathcal{A}_E$. $\mathscr{O}_E$ tosses a bit $b \in \{0,1\}$ and subsequently has the following behavior: When $\mathcal{A}_E$ submits $(N-t)$ chosen plaintexts $(s^h)_{h\in\mathcal{H}}$ to $\mathscr{O}_E$ either $(b = 0)$ then $\mathscr{O}_E$ returns $(N-t)$ encryptions of 0: $(\mathsf{Enc}(\mathsf{pk}_h^{\mathsf{PKE}}, 0))_{h\in\mathcal{H}}$, or: $(b = 1)$ then $\mathscr{O}_E$ returns actual encryptions of the plaintexts $(\mathsf{Enc}(\mathsf{pk}_h^{\mathsf{PKE}}, s^h))_{h\in\mathcal{H}}$.

The reduction is as follows. Upon receiving a set of keys $(\mathsf{pk}_h^{\mathsf{PKE}})_{h\in\mathcal{H}}$ from $\mathscr{O}_E$, then $\mathcal{A}_E$ samples itself t key pairs $(\mathsf{sk}_i^{\mathsf{PKE}}, \mathsf{pk}_i^{\mathsf{PKE}})_{i\in\mathcal{I}}$, initiates $\mathcal{A}_{\mathsf{PSS}}$, reorganizes the indices so that the indices chosen by $\mathcal{A}_{\mathsf{PSS}}$ correspond to $\mathcal{I}$, gives to $\mathcal{A}_{\mathsf{PSS}}$ the total $N = |\mathcal{H}| + |\mathcal{I}|$ public keys and furthermore gives to $\mathcal{A}_E$ the t secret keys $(\mathsf{sk}_i^{\mathsf{PKE}})_{i\in\mathcal{I}}$.

Upon receiving one challenge plaintexts s from $\mathcal{A}_{\mathsf{PSS}}$, $\mathcal{A}_E$ computes the first step of PSS on it, namely: $(s^{(1)}, ..., s^{(N)}) \leftarrow \mathsf{LS}_{R_q}.\mathsf{Share}(s)$. It then sends the challenge $(N-t)$ plaintext messages: $\left(s^{(h)}\right)_{h\in\mathcal{H}}$ to $\mathscr{O}_E$. Upon receiving the response ciphertexts $(c_h)_{h\in\mathcal{H}}$ from $\mathscr{O}_E$, it then computes the N-sized vector V consisting of:

- The entries in $\mathcal{I}$ equal to correct encryptions $\left\{\mathsf{Enc}(\mathsf{pk}_i^{\mathsf{PKE}}, s^{(i)})_{i\in\mathcal{I}}\right\}$ that $\mathcal{A}_E$ generates itself.
- The remaining entries are set to the $\{c_h\}_{h\in\mathcal{H}}$ received from $\mathscr{O}_E$.
  And sends it to $\mathcal{A}_{\mathsf{PSS}}$ as response to its challenge. Upon answer a bit b from $\mathcal{A}_{\mathsf{PSS}}$, then $\mathcal{A}_E$ outputs the same bit b to $\mathscr{O}_E$.

- in the case where the ciphertexts $\{c_h\}_{h\in\mathcal{H}}$ are encryptions of the actual $N-t$ shares $\{s^{(h)}\}_{h\in\mathcal{H}}$, then $\mathcal{A}_{\mathsf{PSS}}$ receives from $\mathcal{A}_E$ a correctly generated PSS of s.

- in the case where the ciphertexts $\{c_h\}_{h\in\mathcal{H}}$ are encryptions of 0, then, by Property 16 of uniform independence of the t plaintext shares $\left(s^{(i)}\right)_{i\in\mathcal{I}}$, we have that what $\mathcal{A}_{\mathsf{PSS}}$ receives from $\mathcal{A}_E$ is undistinguishable from a sample in the distribution $\mathcal{V}$.

Thus in both cases $b \in \{0,1\}$, $\mathcal{A}_{\mathsf{PSS}}$ is faced with the same distribution as would have been generated by oracle $\mathscr{O}_{\mathsf{PSS}}$ for the same b, thus the distinguishing advantage of $\mathcal{A}_E$ is the same as the one of $\mathcal{A}_{\mathsf{PSS}}$.

Handling semi-adaptive security Notice that, although not necessary for our MPC model, where corruptions are done ahead of publications of keys of honest players, one could have imagined a game in which $\mathcal{A}_{\mathsf{PSS}}$ first sees n public keys, then subsequently chooses for which t ones it wants to be revealed the secret keys. However we do not consider corruptions after decryption shares are issued, thus the terminology "semi-adaptive".

We can compile the reduction above to this semi-adaptive setting, although with an exponential loss in n, as follows. $\mathcal{A}_E$, upon having received the $(\mathsf{pk}_h^{\mathsf{PKE}})_{h\in\mathcal{H}}$ from $\mathscr{O}_E$, and sampled itself the t key pairs $(\mathsf{sk}_i^{\mathsf{PKE}}, \mathsf{pk}_i^{\mathsf{PKE}})_{i\in\mathcal{I}}$, shuffles the indices at random. Then it gives the N public keys to $\mathcal{A}_{\mathsf{PSS}}$, which queries t indices of which it wants to be revealed the secret keys. In the case where $\mathcal{A}_E$ would not known at least one secret key of these t out of N indices then $\mathcal{A}_E$ simply outputs a bit b at random to $\mathscr{O}_E$.

Otherwise, it means that $\mathcal{A}_{\mathsf{PSS}}$ queried exactly the t indices which were previously the ones, denoted $\mathcal{I}$, which $\mathcal{A}$ generated itself. Thus in this case $\mathcal{A}_E$ opens the secret keys to $\mathcal{A}_{\mathsf{PSS}}$ and we are exactly in the same situation as in the non-adaptive game. Notice however that this situation happens with probability in $1/\binom{N}{t}$,

D Complements on BFV with Alternative Relinearization

D.1 Generalities

For two polynomials p and d in R_q whose polynomial modulus s a degree-n power of 2 cyclotomic, we have

$$\|pd\| \leq n\|p\|\|d\|. \tag{8}$$

The proof of this inequality is straightforward and it can be found in [BCN18, Lemma 2].

Semi-malicious adversaries are not required to correctly sample in the distributions. They are only required to have values of samples on their witness tapes, which are within the bounds that we gave for the distributions: 1, B, B_{Enc} and B_{sm}. Thus, the definitions and bounds which we now provide are worst cases over the choices of the following:

- $\mathbf{a}$ and $\mathbf{d}_1$, both in R_q^l;
- sk and r, in R_q and both of norm smaller than 1; $\mathbf{e}^{(\mathbf{pk})}$, $\mathbf{e}_0^{(\mathbf{rlk})}$ and $\mathbf{e}_2^{(\mathbf{rlk})}$; in R_q^l and all of norms $\leq B$;

- deduce, from these, an encryption and relinearization keys, namely: $(sk, \mathbf{b})$ and $\mathbf{rlk}$, computed as in $BFV.\mathsf{KeyGen}(\mathbf{a})$ and $\mathsf{CDKS}(\mathbf{a}, \mathbf{d}_1, sk)$.

D.2 Upper-Bound on the Decryption Noise of a Fresh Encryption

We now formalize the set in which belong the outputs of $BFV.\mathsf{Enc}$. For any $m \in R_k$, we denote as a "Fresh BFV Encryption of m", any element of R_q^2 of the form: $ct = (\Delta m + u.\,\mathbf{b}[0] + e_0^{(Enc)}, u\,\mathbf{a}[0] + e_1^{(Enc)})$, where $\|u\| \leq 1$, $\left\|e_0^{(Enc)}\right\| \leq B_{Enc}$ and $\left\|e_1^{(Enc)}\right\| \leq B$. Let us denote $e^{(fresh)} := e^{(Dec)}(ct, sk, m) := ct[0] + ct[1]\,sk - \Delta m$ its decryption noise (Def-Prop 5).

Recall that by definition we have that $ct[0] + ct[1]\,sk = \Delta m + e^{(fresh)}$. With $e^{(pk)} = \mathbf{e}^{(\mathbf{pk})}[0]$, we have

$$\begin{aligned} ct[0] + ct[1]\,sk &= \Delta m + ub + e_0^{(Enc)} + sk\,a\,u + sk\; e_1^{(Enc)} \\ &= \Delta m + (\cancel{-sk\,a} + e^{(pk)})\,u + e_0^{(Enc)} + \cancel{sk\,a\,u} + sk\,e_1^{(Enc)} \\ &= \Delta m \underbrace{+u\,e^{(pk)} + e_0^{(Enc)} + sk\,e_1^{(Enc)}}_{e^{(fresh)}} \end{aligned}$$

$$\left\|e^{(fresh)}\right\| \leq B_{Enc} + n\|e^{(pk)}\| + nB\,\|sk\| := B_{fresh} \tag{9}$$

where $\Delta = \left\lfloor \frac{q}{k} \right\rfloor$.

D.3 Noise Analysis of Addition

Let us consider two ciphertexts ct_1 and ct_2 such that $ct_1[0] + ct_1[1]\,sk = \Delta m_1 + e_1^{(Dec)}$ and $ct_2[0] + ct_2[1]\,sk = \Delta m_2 + e_2^{(Dec)}$. Let $ct_{add} = BFV.Add(ct_1, ct_2)$ be the homomorphic sum of ct_1 and ct_2, and let us define the "decryption noise of an addition" as $e^{(add)} := e^{(Dec)}(ct_{add}, sk, m_1 + m_2)$. Thus we have $ct_{add}[0] + ct_{add}[1]\,sk = \Delta[m_1 + m_2]_k + e^{(add)}$, with $m_1 + m_2 = [m_1 + m_2]_k + k.r$ for $\|r\| \leq 1$ and

$$\|e^{(add)}\| = \|e_1^{(Dec)} + e_2^{(Dec)} + r_k(q)\,r\| \leq \|e_1^{(Dec)}\| + \|e_2^{(Dec)}\| + r_k(q) \tag{10}$$

where $r_k(q)$ denotes the remainder of the integer division of q by k.

D.4 Noise Analysis of Multiplication

Let us consider two ciphertexts ct_1 and ct_2 such that $ct_1[0] + ct_1[0].sk = \Delta m_1 + e_1^{(Dec)}$ and $ct_2[0] + ct_2[1].sk = \Delta m_2 + e_2^{(Dec)}$. Recall from §4.3 that the multiplication of two BFV ciphertexts involves two steps that introduce noise: a *tensoring* operation followed by a *relinearization*.

Tensoring First, let $\hat{ct} = \left\lfloor \frac{k}{q} ct_1 \otimes ct_2 \right\rceil = (\hat{ct}[0], \hat{ct}[1], \hat{ct}[2])$. Let us define the "decryption noise of a three-terms ciphertext $\hat{ct}$ with respect to secret key sk and plaintext $m_1\, m_2$", and denote it $e^{(tens)}$, as:

$$\hat{ct}[0] + \hat{ct}[1]\, sk + \hat{ct}[2]\, sk^2 = \Delta[m_1\, m_2]_k + e^{(tens)} \tag{11}$$

Using [FV12, Lemma 2], we conclude that

$$\|e^{(tens)}\| \leq n\, k\, (\|e_1^{(Dec)}\| + \|e_2^{(Dec)}\|)\, (n\, \|sk\| + 1) + 2k^2\, n^2\, (\|sk\| + 1)^2 \tag{12}$$

which is dominated by the first term. This shows that the noise is roughly multiplied by the factor $2\, k\, n^2\, \|sk\|$.

Relinearization Second, a relinearization is performed using a key, denoted $\mathbf{rlk}$, generated by the CDKS algorithm detailed in §4.3.1. Recall that $\mathbf{rlk} = (\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2)$ where $(\mathbf{d}_0, \mathbf{d}_2) \leftarrow \mathsf{CDKS}(\mathbf{a}, \mathbf{d}_1, sk)$, the latter being defined as: $(\mathbf{e}_0^{(\mathbf{rlk})}, \mathbf{e}_2^{(\mathbf{rlk})}) \leftarrow (\Psi_q^l)^2$ and $r \leftarrow \mathscr{X}_q$; output $(\mathbf{d}_0, \mathbf{d}_2) = \big(-sk.\mathbf{d}_1 + \mathbf{e}_0^{(\mathbf{rlk})} + r.\mathbf{g},\ r.\mathbf{a} + \mathbf{e}_2^{(\mathbf{rlk})} + sk.\mathbf{g}\big)$. Consider a three terms ciphertext $\hat{ct}$ with decryption noise $e^{(tens)}$ with respect to plaintext m ($m_1 m_2$ in our context) and secret key sk. Let us define the relinearized ciphertext of $\hat{ct}$ as $ct' = (ct'[0], ct'[1]) \leftarrow Relin(ct, \mathbf{rlk}, \mathbf{b})$, as defined in §4.3.2, and denote $e^{(relin)}$ the additional decryption noise, namely:

$$\hat{ct}[0] + \hat{ct}[1]\, sk + \hat{ct}[2]\, sk^2 = ct'[0] + ct'[1]\, sk + e^{(relin)} \tag{13}$$

Let us estimate the noise introduced by §4.3.2. Recall that $ct'[2] = \langle \mathbf{g}^{-1}(\hat{ct}[2]), \mathbf{b} \rangle$. Let us denote $err_1 = \left\langle \mathbf{g}^{-1}(ct'[2]), \mathbf{e}_0^{(\mathbf{rlk})} \right\rangle$ and $err_2 = \left\langle \mathbf{g}^{-1}(\hat{ct}[2])\, ,\ sk\, \mathbf{e}^{(\mathbf{pk})} + \mathbf{e}_2^{(\mathbf{rlk})}\, sk \right\rangle$. We have

$$\left\langle \mathbf{g}^{-1}(ct'[2]) \text{<·>} (\mathbf{d}_0, \mathbf{d}_1),\ (1, sk) \right\rangle = r.ct'[2] + \left\langle \mathbf{g}^{-1}(ct'[2]), \mathbf{e}_0^{(\mathbf{rlk})} \right\rangle = r.ct'[2] + err_1 \text{ and}$$

$$\begin{aligned} \langle \mathbf{g}^{-1}(\hat{ct}[2]), \mathbf{d}_2 \rangle\ sk &= \left\langle \mathbf{g}^{-1}(\hat{ct}[2]), -r\, \mathbf{b} + \mathbf{e}^{(\mathbf{pk})}\, sk + \mathbf{e}_2^{(\mathbf{rlk})}\, sk + sk^2.\mathbf{g} \right\rangle \\ &= -r\, ct'[2] + \hat{ct}[2].sk^2 + \left\langle \mathbf{g}^{-1}(\hat{ct}[2]), sk.\mathbf{e}^{(\mathbf{pk})} + \mathbf{e}_2^{(\mathbf{rlk})}.sk \right\rangle \\ &= -r\, ct'[2] + \hat{ct}[2].sk^2 + err_2. \end{aligned}$$

We can now analyze the noise introduced by the relinearization. First, recall that $\mathbf{g}^{-\mathbf{1}}(.)$ decompose an element $x \in R_q$ into a *short* vector $\mathbf{u} = (u_0, ..., u_{l-1}) \in R^l$ such that $\langle \mathbf{u}, \mathbf{g} \rangle = x \mod q$ with $\|u_i\| \leq B_g$ for $i = 0, 1, ..., l-1$. In details,

we can write

$$\begin{aligned} ct'[0] + ct'[1].sk &= \hat{ct}[0] + \mathbf{g}^{-1}(ct^{'}[2])^{\langle\cdot\rangle}(\mathbf{d}_0, \mathbf{d}_1) + \Big(\hat{ct}[1] + \mathbf{g}^{-1}(ct^{'}[2])^{\langle\cdot\rangle}(\mathbf{d}_0, \mathbf{d}_1) \\ &\quad + \left\langle \mathbf{g}^{-1}(\hat{ct}[2]), \mathbf{d}_2 \right\rangle\Big)\, sk \\ &= \hat{ct}[0] + \hat{ct}[1].sk + \left\langle \mathbf{g}^{-1}(ct^{'}[2])^{\langle\cdot\rangle}(\mathbf{d}_0, \mathbf{d}_1), (1, sk) \right\rangle + \left\langle \mathbf{g}^{-1}(\hat{ct}[2]), \mathbf{d}_2 \right\rangle sk \\ &= \hat{ct}[0] + \hat{ct}[1].sk + \hat{ct}[2].sk^2 + err_1 + err_2 \\ &= \Delta m + e^{(tens)} + e^{(relin)} \end{aligned}$$

with $e^{(tens)}$ introduced above and

(14)

$$\|e^{(relin)}\| \leq \|err_1\| + \|err_2\| \leq n\, B_g \|\mathbf{e}_0^{(\mathbf{rlk})}\| + n^2\, l\, B_g \|sk\| (\|\mathbf{e}^{(\mathbf{pk})}\| + \|\mathbf{e}_2^{(\mathbf{rlk})}\|)$$

D.5 How Assumption 6 appears in [CDKS19]

Assumption 6 appears in [CDKS19] with the following notations. They define a RLWE-based symmetric one-time encryption scheme with plaintexts in R_q and ciphertexts in $R_q^{3\times l}$, denoted $UniEnc_{\mathbf{a}}$, parametrized by $\mathbf{a} \in R_q^l$. In their use case, $\mathbf{a} \in R_q^l$ is the URS which is also used to generate $(sk, \mathbf{b}) \leftarrow BFV.\mathsf{KeyGen}(\mathbf{a})$, exactly as in our MPC setting. Then, they state in their (Security) formula p7, and prove in §B.1 that for any (chosen plaintext) μ, we have that: for a sampling $\mathbf{a} \leftarrow U(R_q^l)$, followed by a sampling $(sk, \mathbf{b}) \leftarrow BFV.\mathsf{KeyGen}(\mathbf{a})$, followed by *one single* randomized encryption $UniEnc_{\mathbf{a}}(sk, \mu)$, then the *single output* $(\mathbf{b}, UniEnc_{\mathbf{a}}(sk, \mu))$ is indistinguishable from a single sample in $U(R_q^{l\times 5})$. Next, they assume that (Security) also holds when the chosen μ is replaced by the secret key sk itself, which is exactly what we spelled-out in Assumption 6. Concretely, in their $UniEnc_{\mathbf{a}}$, the r in our $\mathcal{D}_0$ shows up as the secret encryption randomness, while the $\mathbf{d}_1$ is specified in UniEnc to be sampled uniformly when encrypting.

E Practical Parameters Estimation

Recall that the common encryption key generated by $\mathsf{ATFHE.DKG}$ comes as a BFV single-key, of the form $(\mathbf{b}, \mathbf{a}) \in R_q^{2l}$, thus of total bit-size $n.l.logq$. Likewise, the common relinearization key is of the form $(\mathbf{d}_0, \mathbf{d}_1, \mathbf{d}_2) \in R_q^{3l}$, thus of total bit-size $3.n.l.logq$. In the table below we recall the parameters used in [CDKS19, 6.2, Table 2] (which are default settings in the library "SEAL"). Notice that in their table, l is instead denoted "$\#p_i$". The authors indicate that these parameters were chosen following the "the HE security standard" [Alb+21], in order to achieve at least $128bits$ of security. Also, recall that [Alb+21] advises to set $\sigma = 3.2$.

n	$logq$	l
2^{13}	218	4
2^{14}	438	8
2^{15}	881	16

Table 2: Parameters of [CDKS19, 6.2, Table 2], calibrated for at least 128 bits security according to [Alb+21].

F Further Details on the Proof of Theorem 10

F.1 Pseudorandomness of BFV ciphertexts with uniformly generated public keys

First, we want to prove that considering a public key sampled uniformly at random, the ciphertext produced by $BFV.\mathsf{Enc}$ are pseudorandom under the RLWE assumption. The reason is that in the context of Hybrid_3, i.e., in Lemma 12 the view of $\mathscr{E}$ is very similar to the one of the BFV scheme, except that the key is uniformly random. We formalize it by the game *Semantic* shown below:

Setup : The challenger generates samples $\mathbf{a}, \mathbf{b} \leftarrow U(R_q^l)$ and sends $(\mathbf{a}, \mathbf{b})$ to $\mathcal{A}$.

Query : $\mathcal{A}$ chooses a $m \in R_k$ and sends it to the challenger.

Challenge The challenger picks a random $\beta \in \{0, 1\}$.

- If $\beta = 0$, it chooses $ct^* = (c_0^*, c_1^*) \leftarrow R_q^2$ uniformly at random.
- If $\beta = 1$, it generates a valid ciphertext $ct^* = (c_0^*, c_1^*) \leftarrow BFV.\mathsf{Enc}(\mathbf{b}, \mathbf{a}, m)$.

Guess $\mathcal{A}$ gets $ct^* = (c_0^*, c_1^*)$ and outputs $\beta' \in \{0, 1\}$. It wins if $\beta' = \beta$.

Lemma 17. *Let $param = (R_q, l, \mathscr{X}_q, R_k, \Psi_q)$ be parameters such that Assumption 6 holds and $\mathscr{B}_{Enc,q}$ that satisfies Equation 4 in §5.3. Then for any PPT adversary $\mathcal{A}$, the function $AdvCPA_{\mathcal{A}}^{Semantic}(\lambda) := \left|Pr[\beta = \beta'] - \frac{1}{2}\right|$, denoted as the advantage of $\mathcal{A}$, is negligible in λ.*

Proof. In case $\beta = 1$, the adversary is returned the pair $(\Delta m + u.b + e_0^{(Enc)}, a.u + e_1^{(Enc)}) \in R_q^2$, with $b = \mathbf{b}[0]$ and $a = \mathbf{a}[0]$, and where the fixed $u \leftarrow \mathscr{X}_q$, $e_0^{(Enc)} \leftarrow \mathscr{B}_{Enc,q}$ and $e_1^{(Enc)} \leftarrow \Psi_q$ are secretly sampled. Subtracting the known Δm from the left component, the pair constitutes 2 RLWE samples, namely: sample a fixed $u \leftarrow \mathscr{X}_q$, then construct the first RLWE sample with $(b \leftarrow U(R_q), e_0^{(Enc)} \leftarrow \mathscr{B}_{Enc,q})$ and the second one with $(a \leftarrow U(R_q), e_0^{(Enc)} \leftarrow \Psi_q)$. Thus, by RLWE for $(\mathscr{X}_q, \Psi_q)$, and thus a fortiori for $(\mathscr{X}_q, \mathscr{B}_{Enc,q})$ (Equation 4), the two RLWE samples are indistinguishable from a sample in $U(R_q^2)$. □

F.2 IND-CPA under Joint Keys

In [Ash+12, Lemma 3.4], it is proven that an adversary cannot distinguish the ciphertext of a chosen plaintext from a random string, even if the ciphertext

if encrypted under a key of the form $b + b'$, where b' is adaptively generated by the semi-honest adversary after it saw b. Our goal is to adapt their result in the RLWE setting. Since we need only this result in the context of Hybrid_3, i.e., in Lemma 12, we can consider that the honest key b is generated uniformly at random, instead of generated by $BFV.\mathsf{KeyGen}$. We consider an experiment $JointKey(R_q, l, \mathscr{X}_q, R_k, \Psi_q, \mathscr{B}_{Enc,q})$ between an attacker $\mathcal{A}$ and a challenger defined as:

Setup The challenger generates samples $\mathbf{a}, \mathbf{b} \leftarrow U(R_q^l)$ and sends $(\mathbf{a}, \mathbf{b})$ to $\mathcal{A}$.

Query $\mathcal{A}$ adaptively chooses: t pairs $(sk_i, \mathbf{e}_i^{(\mathbf{pk})})_{i\in\mathcal{I}}$, both terms being either $\perp$ or such that $\|sk_i\| = 1$ and $\|\mathbf{e}_i^{(\mathbf{pk})}\| \leq l\,B$. Define $sk' := \Sigma_{i\in\mathcal{I}} sk_i$ where the $\perp$ values are set to 0, and likewise for $\mathbf{e}^{(\mathbf{pk})} := \sum_{i\in\mathcal{I}} \mathbf{e}_i^{(\mathbf{pk})}$. $\mathcal{A}$ outputs $\left\{\mathbf{b}' = -\mathbf{a}.sk' + \mathbf{e}^{(\mathbf{pk})}, (sk'_i)_{i\in\mathcal{I}}, (\mathbf{e}_i^{(\mathbf{pk})})_{i\in\mathcal{I}}\right\}$ to the challenger, along with some $m \in R_k$ of its choice.

Challenge The challenger sets the $\mathbf{pk} = \mathbf{b} + \mathbf{b}'$ and picks a random $\beta \in \{0, 1\}$.

- If $\beta = 0$, it chooses $c^* = (c_0^*, c_1^*) \leftarrow R_q^2$ uniformly at random.
- If $\beta = 1$, it generates a valid ciphertext $c^* = (c_0^*, c_1^*) \leftarrow BFV.\mathsf{Enc}(\mathbf{pk}, \mathbf{a}, m)$.

Guess $\mathcal{A}$ gets $c^* = (c_0^*, c_1^*)$ and outputs $\beta' \in \{0, 1\}$. It wins if $\beta' = \beta$.

Lemma 18. *Let $param = (R_q, l, \mathscr{X}_q, R_k, \Psi_q)$ be parameters such that Assumption 6 holds and $\mathscr{B}_{Enc,q}$ that satisfies Equation 4 in §5.3. Then for any PPT adversary $\mathcal{A}$, we have:*

$$Pr\big[JointKey_{\mathcal{A}}(R_q, l, \mathscr{X}_q, R_k, \Psi_q, \mathscr{B}_{Enc,q}) = 1\big] - 1/2 = negl(\lambda). \tag{15}$$

Proof. We construct an adversary $\mathcal{A}'$ playing the game of Lemma 17. $\mathcal{A}'$ uses as black box an adversary $\mathcal{A}$ for $JointKey(R_q, l, \mathscr{X}_q, R_k, \Psi_q, \mathscr{B}_{Enc,q})$, as follows. The challenger gives $\mathcal{A}'$ the value $\mathbf{b}$, and a ciphertext (c_0, c_1) which is either chosen as $BFV.\mathsf{Enc}(\mathbf{b}, \mathbf{a}, 0)$ ($\beta = 1$) or is a sample in $U(R_q^2)$ ($\beta = 0$). Then $\mathcal{A}'$ gives $\mathbf{b}$ to $\mathcal{A}$ and gets back $\big(\mathbf{b}' = -\mathbf{a}.sk' + \mathbf{e}^{(\mathbf{pk})}, sk', \mathbf{e}^{(\mathbf{pk})}, m\big)$ from $\mathcal{A}$, where m is a challenge plaintext. Finally, $\mathcal{A}'$ sets $(c_0^*, c_1^*) = (c_0 - c_1.sk', c_1) \in R_q^2$, sends it to $\mathcal{A}$ and outputs the bit β' obtained from $\mathcal{A}$.

It is easy to see that if $\beta = 0$, then (c_0^*, c_1^*) is uniformly random. On the other hand, if $\beta = 1$, we can write $c_0 = u.b + e_0^{(Enc)} \in R_q$ and $c_1^* = u.a + e_1^{(Enc)} \in R_q$ for some $u \leftarrow \mathscr{X}_q$, $e_0^{(Enc)} \leftarrow \mathscr{B}_{Enc,q}$, $e_1^{(Enc)} \leftarrow \Psi_q$ and $b = \mathbf{b}[0]$, $a = \mathbf{a}[0]$, and with $e^{(pk)} = \mathbf{e}^{(\mathbf{pk})}[0]$:

$$\begin{aligned} c_0^* &= u.b + e_0^{(Enc)} - c_1.sk' = u.b + e_0^{(Enc)} - (u.a + e_1^{(Enc)}).sk' \\ &= u(b + b') + e_0^{(Enc)} - e_1^{(Enc)}.sk' - u.e^{(pk)} \\ &\overset{s}{\equiv} u.(b + b') + e_0^{(Enc)} \end{aligned}$$

The last equality states a statistical indistinguishability between the distributions of $e_0^{(Enc)} - e_1^{(Enc)}.sk' - u.e^{(pk)}$ and of $e_0^{(Enc)}$, which we now prove. To start with, from equation (8), we have both $\|e_1^{(Enc)}.sk'\| \leq nNB$ and $\|u.e^{(pk)}\| \leq nNB$. Thus, $\|e_1^{(Enc)}.sk' - u.e^{(pk)}\| \leq 2nNB$. But on the other hand, $\|e_0^{(Enc)}\| \leq$

B_{Enc}. We conclude since the parameters are chosen such that $\frac{2nNB}{B_{Enc}} = negl(\lambda)$ (cf Equation (4) in §5.2.4). This conclusion can be formalized as the "smudging Lemma 19" below, which implies that, in the sum $e_0^{(Enc)} - e_1^{(Enc)}.sk' - u.e^{(pk)}$, we have that the distribution of $-e_1^{(Enc)}.sk' - u.e^{(pk)}$ is "smudged-out" by the one of $e_0^{(Enc)}$. is an instance of RLWE problem. Therefore, $\mathcal{A}'$ acts indistinguishably from the challenger of the Game of Lemma 17, thus has the same advantage as $\mathcal{A}$. □

Lemma 19 (Smudging lemma). *Adopting the notations introduced in §4.1, let R be a ring of dimension n. Let $\mathscr{B}_1$ be a Gaussian distribution over R of variance σ_1^2. Following [Kim+20], one can specify an interval $[-B_1, B_1] = [-6\,\sigma_1\sqrt{n}, 6\,\sigma_1\sqrt{n}]$ such that, for $e_1 \leftarrow \mathscr{B}_1$, $\|e_1\| \leq B_1$ with high probability. Similarly, let $e_2 \leftarrow \mathscr{B}_2$ be sampled from another Gaussian distribution $\mathscr{B}_2$ of variance σ_2^2. Then the distribution of e_2 is statistically indistinguishable from that of $e_2 + e_1$ if $\sigma_1^2/\sigma_2^2 = \epsilon$, where $\epsilon = \epsilon(\lambda)$ is a negligible function.*

Proof. For e_1 and e_2 that follow centered Gaussian distributions $\mathscr{B}_1$ and $\mathscr{B}_2$ of different variances denoted σ_1^2 and σ_2^2 respectively, if the ratio σ_1^2/σ_2^2 is negligible in λ, then $e_1 + e_2$ is statistically indistinguishable from e_2 from Smudging Lemma [Ash+12]. □

F.3 Full Details of Hybrids of the Proof of Theorem 10

In section §5.4, we detailed a simulator $\mathcal{S}$ such that no PPT environment $\mathscr{E}$, which can choose the honest inputs, observe the honest outputs, and fully controls t out of the $N = 2t + 1$ players (via an adversary $\mathcal{A}$), can distinguish between: (i) the real protocol $REAL_{\Pi_C^{\mathcal{F}_{\text{DLC}}}}$, where $\mathscr{E}$ interacts with the adversary $\mathcal{A}$, and honest players follow the actual protocol $\Pi_C^{\mathcal{F}_{\text{DLC}}}$, and (ii) the ideal protocol $IDEAL_{\mathcal{F}_C,\mathcal{S},\mathscr{E}}$, where $\mathscr{E}$ interacts with $\mathcal{S}$, while honest players are connected only to $\mathcal{F}_C$

We now spell out in full details the series of hybrid games used in §5.4 to prove the indistinguishability of the real and ideal worlds. The output of each game is the output of the environment. Changes with respect to the previous Hybrid are highlighted in blue. We abuse notations in that when we denote that "players perform some action", we mean implicitely that dishonest players perform it in accordance with a semi-Malicious behavior (§2.4), i.e., send part of, or none, of the messages instructed, and, when sending some, arbitrarily select their random parameters, as long as they are within the essential bounds of the prescribed distributions, i.e., B, B_{sm}, B_{Enc}.

Hybrid_1*:*

- **Setup:** Each player $P_i \in \mathscr{P}$ does the following:
  (0) Send (Setup) to $\mathcal{F}_{\text{DLC}}$.
  (0) Run $(\mathbf{a}, \mathbf{d}_1) \leftarrow \mathsf{ATFHE.DKG.Setup}()$.

- **Input distribution:** Upon ready from $\mathcal{F}_{\text{DLC}}$, each input owner $\ell \in \mathscr{L}$:
  (1) Runs $\mathsf{ATFHE.Enc}(m_\ell)$ then go offline.
- **Key distribution:** Upon ready from $\mathcal{F}_{\text{DLC}}$, each player P_i:
  (1) Computes $\mathbf{b}_i \leftarrow \mathsf{ATFHE.DKG.Contrib}(\mathbf{a})$ and $(\mathbf{d}_{0,i}, \mathbf{d}_{2,i}) \leftarrow \mathsf{ATFHE.DRlkG.Contrib}(\mathbf{a}, \mathbf{d}_1, sk_i)$.
  (1) Runs $\mathsf{ATFHE.DNG.Contrib}()$
  (1) Sends $\mathbf{b}_i, (\mathbf{d}_{0,i}, \mathbf{d}_{2,i})$ over BC^{P_i}.

As in the protocol, we denote $S_{ct} \subset \mathscr{L}$ the owners, resp. $S \subset \mathscr{P}$ the players, for which no instance returned $\perp$.

- **Transformation and evaluation:** Each player P_i:
  (2) $\forall j \in S$, parses the outputs of BC^{P_j} as $(\mathbf{b}_j, (\mathbf{d}_{0,j}, \mathbf{d}_{2,j}))$.
  (2) Compute $\mathbf{b} \leftarrow \mathsf{ATFHE.DKG.Combine}(\{\mathbf{b}_j\}_{j\in S})$ and $\mathbf{rlk} \leftarrow \mathsf{ATFHE.DRlkG.Combine}((\mathbf{d}_{0,j}, \mathbf{d}_{2,j})_{j\in S}, \mathbf{d}_1)$.
  (2) Run, $\forall \ell \in S_{ct}$, $\mathsf{ATFHE.Transform}((\overline{\Delta m_\ell}, \overline{u_\ell}, \overline{e_{0,\ell}^{(Enc)}}, \overline{e_{1,\ell}^{(Enc)}}), \mathbf{b}, \mathbf{a})$.
  (3) Upon receiving $(\Lambda_{Enc}^{a,b}, ct_j)$ for all $j \in S_{ct}$ from $\mathcal{F}_{\text{DLC}}$, computes $ct \leftarrow BFV.\mathsf{Eval}(C', \{ct_j\}_{j\in S_{ct}}, \mathbf{rlk}, \mathbf{b})$.
  (3) Run $\mathsf{ATFHE.PartDec}(ct, \{\overline{\eta_i}, \overline{sk_i}\}_{i\in S})$.
- **Output computation:** Denote $y := C'(\{m_\ell\}_{\ell\in S_{ct}})$
  - $\mathcal{F}_{\text{DLC}}$ delay-outputs $\left(\Lambda_{dec}^{ct},\ \mu^{\mathcal{S}} := \Delta.y + \Sigma_{j\in S}\eta_j\right)$ to all players. Upon receiving it from $\mathcal{F}_{\text{DLC}}$, players output $m \leftarrow \mathsf{ATFHE.FinDec}(\mu^{\mathcal{S}})$.

Hybrid_2*:*

- **Setup:** Each player P_i:
  (0) Send (Setup) to $\mathcal{F}_{\text{DLC}}$.
  (0) Run $(\mathbf{a}, \mathbf{d}_1) \leftarrow \mathsf{ATFHE.DKG.Setup}()$.
- **Input distribution:** Upon ready from $\mathcal{F}_{\text{DLC}}$, each input owner $\ell \in \mathscr{L}$:
  (1) Runs $\mathsf{ATFHE.Enc}(m_\ell)$ then go offline.
- **Key distribution:** Upon ready from $\mathcal{F}_{\text{DLC}}$, each player P_i:
  (1) *If P_i corrupt*, then unchanged instructions: Compute $\mathbf{b}_i \leftarrow \mathsf{ATFHE.DKG.Contrib}(\mathbf{a})$ and $(\mathbf{d}_{0,i}, \mathbf{d}_{2,i}) \leftarrow \mathsf{ATFHE.DRlkG.Contrib}(\mathbf{a}, \mathbf{d}_1, sk_i)$.
  (1) *If P_i honest:* Samples $sk_i \leftarrow \mathscr{X}_q$ and sends (input, sk_i) to $\mathcal{F}_{\text{DLC}}$. Computes $\mathbf{b}_i \leftarrow U(R_q^l)$ and $(\mathbf{d}_{0,i}, \mathbf{d}_{2,i}) \leftarrow U(R_q^{2\times l})$.
  (1) Runs $\mathsf{ATFHE.DNG.Contrib}()$

(1) Sends $\mathbf{b}_i, (\mathbf{d}_{0,i}, \mathbf{d}_{2,i})$ over BC^{P_i}.

As in the protocol, we denote $S_{ct} \subset \mathscr{L}$ the owners, resp. $S \subset \mathscr{P}$ the players, for which no instance returned $\perp$.

- **Transformation and evaluation:** Each player P_i:
  (2) $\forall j \in S$, parses the outputs of BC^{P_j} as $(\mathbf{b}_j, (\mathbf{d}_{0,j}, \mathbf{d}_{2,j}))$.
  (2) Compute $\mathbf{b} \leftarrow \mathsf{ATFHE.DKG.Combine}(\{\mathbf{b}_j\}_{j\in S})$ and $\mathbf{rlk} \leftarrow \mathsf{ATFHE.DRlkG.Combine}\big((\mathbf{d}_{0,j}, \mathbf{d}_{2,j})_{j\in S}), \mathbf{d}_1\big)$.
  (2) Run, $\forall \ell \in S_{ct}$, $\mathsf{ATFHE.Transform}((\overline{\Delta m_\ell}, \overline{u_\ell}, \overline{e_{0,\ell}^{(Enc)}}, \overline{e_{1,\ell}^{(Enc)}}), \mathbf{b}, \mathbf{a})$.
  (3) Upon receiving $(\Lambda_{Enc}^{a,b}, ct_j)$ for all $j \in S_{ct}$ from $\mathcal{F}_{\mathrm{DLC}}$, computes $ct \leftarrow BFV.\mathsf{Eval}(C', \{ct_j\}_{j\in S_{ct}}, \mathbf{rlk}, \mathbf{b})$.
  (3) Run $\mathsf{ATFHE.PartDec}(ct, \{\overline{\eta_i}, \overline{sk_i}\}_{i\in S})$.
- **Output computation:** Each player P_i:
  - $\mathcal{F}_{\mathrm{DLC}}$ delay-outputs $\left(\Lambda_{dec}^{ct},\ \mu^{\mathcal{S}} := \Delta.y + \Sigma_{j\in S}\eta_j\right)$ to all players. Upon receiving it from $\mathcal{F}_{\mathrm{DLC}}$, players output $m \leftarrow \mathsf{ATFHE.FinDec}(\mu^{\mathcal{S}})$.

Hybrid_3*:*

- **Setup:** Each player P_i does the following:
  (0) Send (Setup) to $\mathcal{F}_{\mathrm{DLC}}$.
  (0) Run $(\mathbf{a}, \mathbf{d}_1) \leftarrow \mathsf{ATFHE.DKG.Setup}()$.
- **Input distribution:** Upon ready from $\mathcal{F}_{\mathrm{DLC}}$, every input owner $\ell \in \mathscr{L}$:
  (1) *If ℓ corrupt*: instructions unchanged, i.e., runs $\mathsf{ATFHE}.Enc(\widetilde{m_\ell} := m_\ell)$.
  (1) *If ℓ honest*: Sets $\widetilde{m_\ell} := 0$ and runs $\mathsf{ATFHE}.Enc(\widetilde{m_\ell})$.
- **Key distribution:** Upon ready from $\mathcal{F}_{\mathrm{DLC}}$, each player P_i:
  (1) *If P_i corrupt*: Compute $\mathbf{b}_i \leftarrow \mathsf{ATFHE.DKG.Contrib}(\mathbf{a})$ and $(\mathbf{d}_{0,i}, \mathbf{d}_{2,i}) \leftarrow \mathsf{ATFHE.DRlkG.Contrib}(\mathbf{a}, \mathbf{d}_1, sk_i)$.
  (1) *If P_i honest:* Samples $sk_i \leftarrow \mathscr{X}_q$ and sends (input, sk_i) to $\mathcal{F}_{\mathrm{DLC}}$. Computes $\mathbf{b}_i \leftarrow U(R_q^l)$ and $(\mathbf{d}_{0,i}, \mathbf{d}_{2,i}) \leftarrow U(R_q^{2\times l})$.
  (1) Runs $\mathsf{ATFHE.DNG.Contrib}()$
  (1) Sends $\mathbf{b}_i$, $(\mathbf{d}_{0,i}, \mathbf{d}_{2,i})$ over BC^{P_i}.

As in the protocol, we denote $S_{ct} \subset \mathscr{L}$ the owners, resp. $S \subset \mathscr{P}$ the players, for which no instance returned $\perp$.

- **Transformation and evaluation:** Each player P_i:
  (2) $\forall j \in S$, parses the outputs of BC^{P_j} as $(\mathbf{b}_j, (\mathbf{d}_{0,j}, \mathbf{d}_{2,j}))$.

(2) Compute $\mathbf{b} \leftarrow \mathsf{ATFHE.DKG.Combine}(\{\mathbf{b}_j\}_{j\in S})$ and $\mathbf{rlk} \leftarrow \mathsf{ATFHE.DRlkG.Combine}((\mathbf{d}_{0,j}, \mathbf{d}_{2,j})_{j\in S}, \mathbf{d}_1)$.

(2) Run, $\forall \ell \in S_{ct}$, $\mathsf{ATFHE.Transform}((\overline{\Delta m_\ell}, \overline{u_\ell}, \overline{e_{0,\ell}^{(Enc)}}, \overline{e_{1,\ell}^{(Enc)}}), \mathbf{b}, \mathbf{a})$.

(3) Upon receiving $(\Lambda_{Enc}^{a,b}, \widetilde{ct_j})$ for all $j \in S_{ct}$ from $\mathcal{F}_{\mathrm{DLC}}$, computes $\widetilde{ct} \leftarrow BFV.\mathsf{Eval}(C', \{\widetilde{ct_j}\}_{j\in S_{ct}}, \mathbf{rlk}, \mathbf{b})$.

(3) Run $\mathsf{ATFHE.PartDec}(\widetilde{ct}, \{\overline{\eta_i}, \overline{sk_i}\}_{i\in S})$.

- **Output computation:** Each player P_i:
  - $\mathcal{F}_{\mathrm{DLC}}$ delay-outputs $\left(\Lambda_{dec}^{\widetilde{ct}},\ \mu^{\mathcal{S}} := \Delta.y + \Sigma_{j\in S}\eta_j\right)$ to all players. Upon receiving it from $\mathcal{F}_{\mathrm{DLC}}$, players output $m \leftarrow \mathsf{ATFHE.FinDec}(\mu^{\mathcal{S}})$.

G Bootstrapping

Similar to what we did for the relinearization, we follow the bootstrapping of [CDKS19] for multikey FHE and particularize it in our single-key context. In short, they follow the algorithm improved by [CH18], that consists in four steps, denoted as (1) Modulus raise, (2) Linear Transformation, (3) Extraction and (4) the Inverse Linear Transformation.

Technically, the linear transformation requires the homomorphic evaluation of the rotation of plaintext slots. In addition [CDKS19] also require the homomorphic evaluation of "Galois elements slot-by-slot". In [GHS12b, p23] it is explained how these two evaluations, i.e., operations on plaintexts, can be decomposed into additions, scalar multiplications and applications of automorphisms of $R_k = \mathbb{Z}_k[X]/(X^n+1)$, denoted $\{\tau_j,\ j \in (\mathbb{Z}/2n)^*\}$, each being defined by: $X \to X^j$. In [CDKS19], it is observed that homomorphic evaluation of each τ_j can be realized with the auxiliary key, which we denoted as $\left(\mathbf{h}_0(j),\ \mathbf{h}_1\right)$ whose technical purpose is "key-switching".

In some more details, given a ciphertext $ct = (ct[0], ct[1]) \in R_q^2$ of m, the goal is to homomorphically evaluate τ_j on the plaintext, i.e. to find ct' such that $\langle ct', s\rangle = \tau_j(\langle ct, s\rangle)$. To achieve this, we first compute $\tau_j(ct) = (\tau_j(ct[0]), \tau_j(ct[1]))$ the ciphertext obtained by taking τ_j to the entries of ct. Then $\tau_j(ct)$ is a valid encryption of $\tau_j(m)$ corresponding the secret key $\tau_j(sk)$. The key-switching procedure is then applied to $\tau_j(ct)$, which has for consequence to generate a new ciphertext encrypting the same message under the original secret key s instead of $\tau_j(s)$. This key switching algorithm presented below is adapted from [CDKS19].

G.1 Key Switching algorithm (adapted from [CDKS19])

We now provide the key switching method $KeySwitch$ and then discuss its correctness. It takes as input $ct = (ct[0], ct[1]) \in R_q^2, \mathbf{bk} = [\mathbf{h}_0|\mathbf{h}_1] \in (R_q^l)^2$, and outputs $ct' = (ct'[0], ct'[1]) \in R_q^2$ as follows:

1 $ct'[0] \leftarrow \tau_j(ct[0])$

2 $ct^{'}[0] \leftarrow ct^{'}[0] + \left\langle \mathbf{g}^{-1}(\tau_j(ct[1])), \mathbf{h}_0 \right\rangle$
3 $ct^{'}[1] \leftarrow \left\langle \mathbf{g}^{-1}(\tau_j(ct[1])), \mathbf{h}_1 \right\rangle$

Correctness From the definition and with $s = (1, sk)$, the output ciphertext $ct' = (ct^{'}[0], ct^{'}[1]) \leftarrow KeySwitch(ct, \mathbf{bk})$ holds

$$\begin{aligned} ct^{'}[0] + ct^{'}[1]\, sk &= \tau_j(ct[0]) + \left\langle \mathbf{g}^{-1}(\tau_j(ct[1])), \mathbf{h}_0 \right\rangle + \left\langle \mathbf{g}^{-1}(\tau_j(ct[1])), \mathbf{h}_1) \right\rangle \\ &\approx \tau_j(ct[0]) + \left\langle \mathbf{g}^{-1}(\tau_j(ct[1])), \tau_j(s).\mathbf{g} \right\rangle = \langle \tau_j(ct), \tau_j(sk) \rangle = \tau_j(\langle ct, s \rangle) \end{aligned}$$

as desired.

H Instantiation of ATFHE with Π_{DLC}

- $\mathsf{ATFHE.DKG.Setup}()$: Obtain common uniform strings $(\mathbf{a}, \mathbf{d}_1) \leftarrow \overline{\mathcal{G}}_{\mathrm{URS}}$, with $\mathbf{a} \in R_q^l$ and $\mathbf{d}_1 \in R_q^l$.
- $\mathsf{DLC.Setup}(1^\lambda)$: Compute $(\mathsf{pk}^{\mathsf{PKE}}, \mathsf{sk}^{\mathsf{PKE}}) \leftarrow \mathsf{KeyGen}(1^\lambda)$.
- $\mathsf{ATFHE.Enc}(m \in R_k, \{\mathsf{pk}_i^{\mathsf{PKE}}\}_{i\in[N]})$: Sample $u \leftarrow \mathscr{X}_q$, $e_0^{(Enc)} \leftarrow \mathscr{B}_{Enc,q}$ and $e_1^{(Enc)} \leftarrow \Psi_q$, and compute $(pss_m, pss_u, pss_{e_0^{(Enc)}}, pss_{e_1^{(Enc)}}) = PSS(m, u, e_0^{(Enc)}, e_1^{(Enc)}, \{\mathsf{pk}_i^{\mathsf{PKE}}\}_{i\in[N]})$. Output $(pss_m, pss_u, pss_{e_0^{(Enc)}}, pss_{e_1^{(Enc)}})$.
- $\mathsf{ATFHE.DKG.Contrib}(\mathbf{a}, \{\mathsf{pk}_i^{\mathsf{PKE}}\}_{i\in[N]})$: Sample a noise contribution $\mathbf{e}_i^{(\mathbf{pk})} \leftarrow \Psi_q^l$ and sample a secret key contribution $sk_i \leftarrow \mathscr{X}_q$. Compute $pss_{sk_i} = PSS(sk_i, \{\mathsf{pk}_i^{\mathsf{PKE}}\}_{i\in[N]})$. Output $(\mathbf{b}_i = -sk_i\,\mathbf{a} + \mathbf{e}_i^{(\mathbf{pk})}, pss_{sk_i})$.
- $\mathsf{ATFHE.DNG.Contrib}$ $(\{\mathsf{pk}_i^{\mathsf{PKE}}\}_{i\in[N]})$: Sample $\eta \leftarrow \mathscr{B}_{sm}$ and compute $pss_\eta = PSS(\eta, \{\mathsf{pk}_i^{\mathsf{PKE}}\}_{i\in[N]})$. Output pss_η.
- $\mathsf{ATFHE.DRlkG.Contrib}(\mathbf{a}, \mathbf{d}_1, sk_i)$: Compute $(\mathbf{d}_{0,i}, \mathbf{d}_{2,i}) \leftarrow \mathsf{CDKS}(\mathbf{a}, \mathbf{d}_1, sk_i)$.
- $\mathsf{ATFHE.BkDKG.Contrib}(\mathbf{h}_1, \Psi_q, l, sk_i)$ [Parameter j implicit]: Given a secret key contribution $sk_i \in R_q$, sample $\mathbf{e}^{(\mathbf{bk})} \leftarrow \Psi_q^l$ and compute $\mathbf{h}_{0,i} = -sk_i\,\mathbf{h}_1 + \mathbf{e}^{(\mathbf{bk})} + \tau_j(sk_i)\,\mathbf{g}$.

Transformation

- $\mathsf{ATFHE.DKG.Combine}\big(\{\mathbf{b}_i \in R_q^l\}_{i\in S\subset[N]}\big)$: Output $\mathbf{b} = \Sigma_{i\in S}\mathbf{b_i}$.
- $\mathsf{ATFHE.PartTrans}((pss_m, pss_u, pss_{e_0^{(Enc)}}, pss_{e_1^{(Enc)}}), \mathbf{b}, \mathbf{a}, \mathsf{sk}_j^{\mathsf{PKE}})$: Given an *ad-hoc ciphertext* $(pss_m, pss_u, pss_{e_0^{(Enc)}}, pss_{e_1^{(Enc)}})$ and key $\mathbf{b}$, parse $a := \mathbf{a}[0]$ and $b := \mathbf{b}[0]$, then for $pss_m = (c_m^{(1)}, ..., c_m^{(N)})$ (resp $u, e_0^{(Enc)}, e_1^{(Enc)}$), where $c_{m^{(i)}} = \mathsf{Enc}(pk_i^{\mathsf{PKE}}, m^{(i)})$, compute $m^{(j)} = \mathsf{PKE.Dec}(\mathsf{sk}_j^{\mathsf{PKE}}, c_m^{(j)})$ (resp $u, e_0^{(Enc)}, e_1^{(Enc)}$).

  Output $ct^{(j)} = \Lambda_{Enc}^{b,a}(\Delta m^{(j)}, u^{(j)}, e_0^{(Enc,(j))}, e_1^{(Enc,(j))})$.

- $\mathsf{ATFHE.FinTrans}(\{ct^{(j)}\}_{j\in\mathcal{U}})$: Upon receiving a set $\{ct^{(j)}\}_{j\in\mathcal{U}}$ from any $(t+1)$-set $\mathcal{U}\subset[N]$, compute $ct := \mathsf{LS}_{R_q}.\mathsf{SReco}^{\mathcal{U}}([ct^{(j)},\ j\in\mathcal{U}])$. Output ct.
- $\mathsf{ATFHE.DRlkG.Combine}((\mathbf{d}_{0,i},\mathbf{d}_{2,i})_{i\in S},\mathbf{d}_1)$: output

$$\mathbf{rlk} = (\Sigma_{j\in S}\mathbf{d}_{0,i},\mathbf{d}_1,\Sigma_{j\in S}\mathbf{d}_{2,i}). \tag{16}$$

- $\mathsf{ATFHE.BkDKG.Combine}(\mathbf{h}_1,\{\mathbf{h}_{0,i}\}_{i\in S})$ [Parameter j implicit]: Given a set $\{\mathbf{h}_{1,i}\}_{i\in S}$, compute

$$\mathbf{bk} = (\Sigma_{i\in S}\mathbf{h}_{0,i},\mathbf{h}_1) \tag{17}$$

Distributed decryption

- $\mathsf{ATFHE.PartDec}(ct,\{pss_{\eta_i},pss_{sk_i}\}_{i\in S},\mathsf{sk}_j^{\mathsf{PKE}})$: Given a ciphertext $ct\in R_q^2$ a set $\{pss_{\eta_i},pss_{sk_i}\}_{i\in S}$, with $pss_{\eta_i} = (c_{\eta_i}^{(1)},...,c_{\eta_i}^{(N)})$ (resp sk_i), compute $\eta_i^{(j)} = \mathsf{PKE.Dec}(\mathsf{sk}_j^{\mathsf{PKE}},c_{\eta_i}^{(j)})$. Output $p^{(j)} = \Lambda_{Dec}^{ct}(\{sk_i^{(j)},\eta_i^{(j)}\}_{i\in S}) := ct[0] + ct[1].\Sigma_{i\in S}sk_i^{(j)} + \Sigma_{i\in S}\eta_i^{(j)}$.
- $\mathsf{ATFHE.FinDec}(\{p^{(j)}\}_{j\in\mathcal{U}})$: Upon receiving a set $\{p^{(j)}\}_{j\in\mathcal{U}}$ from any $(t+1)$-set $\mathcal{U}\subset[N]$, compute $y := \mathsf{LS}_{R_q}.\mathsf{SReco}^{\mathcal{U}}([p^{(j)},\ j\in\mathcal{U}])$. Output $\lfloor(k/q).y\rceil \pmod k$.

References for the Appendices.

[Abh21] P. A. and Abhishek Jain and Zhengzhong Jin and Giulio Malavolta. "Unbounded Multi-Party Computation from Learning with Errors". In: *EUROCRYPT*. 2021.

[Agg16] J. A. G. and Aggelos Kiayias and Nikos Leonardos and Giorgos Panagiotakos. "Bootstrapping the Blockchain, with Applications to Consensus and Fast PKI Setup". In: *PKC*. 2016.

[AM69] M. F. Atiyah and I. G. MacDonald. *Introduction To Commutative Algebra*. 1969.

[Ana+20] P. Ananth, A. R. Choudhuri, A. Goel, and A. Jain. "Towards Efficiency-Preserving Round Compression in MPC". In: *ASIACRYPT*. 2020.

[ANRX21] I. Abraham, K. Nayak, L. Ren, and Z. Xiang. "Good-case Latency of Byzantine Broadcast: a Complete Categorization". In: *PODC*. 2021.

[Bad+21] S. Badrinarayanan, P. Miao, P. Mukherjee, and D. Ravi. *On the Round Complexity of Fully Secure Solitary MPC with Honest Majority*. Cryptology ePrint Archive, Report 2021/241. `https://ia.cr/2021/241`. 2021.

[BCG21] E. Boyle, R. Cohen, and A. Goel. "Breaking the $O(\sqrt{n})$-Bit Barrier: Byzantine Agreement with Polylog Bits Per Party". In: *PODC*. 2021.

[BCN18] C. Boschini, J. Camenisch, and G. Neven. "Relaxed lattice-based signatures with short zero-knowledge proofs". In: *International Conference on Information Security*. Springer. 2018, pp. 3–22.

[BHZ21] C. Badertscher, J. Hesse, and V. Zikas. "On the (Ir)Replaceability of Global Setups, or How (Not) to Use a Global Ledger". In: *TCC*. 2021.

[Bor96] M. Borcherding. "Levels of authentication in distributed agreement". In: *WDAG*. 1996.

[BS20] D. Boneh and V. Shoup. *A Graduate Course in Applied Cryptography*. Version 0.5 Jan 2020. 2020.

[Cam+16] J. Camenisch, R. R. Enderlein, S. Krenn, R. Küsters, and D. Rausch. "Universal Composition with Responsive Environments". In: *ASIACRYPT*. 2016.

[Can01] R. Canetti. "Universally composable security: A new paradigm for cryptographic protocols". In: *FOCS*. We refer to eprint 2000/067 version 02/20/2020. 2001.

[Can04] R. Canetti. "Universally Composable Signature, Certification, and Authentication". In: *CSFW*. IEEE Computer Society, 2004, p. 219.

[CD20] I. Cascudo and B. David. "ALBATROSS: Publicly AttestabLe BATched Randomness Based On Secret Sharing". In: *ASIACRYPT*. 2020.

[CDN15] R. Cramer, I. Damgård, and J. Nielsen. *Secure Multiparty Computation and Secret Sharing*. Cambridge University Press, 2015.

[CGGM00] R. Canetti, O. Goldreich, S. Goldwasser, and S. Micali. "Resettable Zero-Knowledge (Extended Abstract)". In: *STOC*. 2000.

[Che+17] J. H. Cheon, A. Kim, M. Kim, and Y. Song. "Homomorphic encryption for arithmetic of approximate numbers". In: *ASIACRYPT*. 2017.

[Che+20] H. Chen, M. Kim, I. P. Razenshteyn, D. Rotaru, Y. Song, and S. Wagh. "Maliciously Secure Matrix Multiplication with Applications to Private Deep Learning". In: *ASIACRYPT*. 2020.

[CLOS02] R. Canetti, Y. Lindell, R. Ostrovsky, and A. Sahai. "Universally Composable Two-Party and Multi-Party Secure Computation". In: *STOC*. 2002.

[Coh16] R. Cohen. "Asynchronous Secure Multiparty Computation in Constant Time". In: *PKC*. 2016.

[Dac+21] D. Dachman-Soled, H. Gong, M. Kulkarni, and A. Shahverdi. "Towards a Ring Analogue of the Leftover Hash Lemma". In: *Journal of Mathematical Cryptology* (2021).

[Dam+09] I. Damgård, M. Geisler, M. Kroigaard, and J. Nielsen. "Asynchronous multiparty computation: Theory and Implementation". In: *PKC*. 2009.

[Daz+08] V. Daza, J. Herranz, P. Morillo, and C. Ràfols. "Ad-Hoc Threshold Broadcast Encryption with Shorter Ciphertexts". In: *Electron. Notes Theor. Comput. Sci.* (2008).

[DDOPS01] A. De Santis, G. Di Crescenzo, R. Ostrovsky, G. Persiano, and A. Sahai. "Robust non-interactive zero knowledge". In: *CRYPTO 2001*. 2001.

[DLS88] C. Dwork, N. Lynch, and L. Stockmeyer. "Consensus in the Presence of Partial Synchrony". In: *J. ACM* (1988).

[FN09] M. Fitzi and J. B. Nielsen. "On the Number of Synchronous Rounds Sufficient for Authenticated Byzantine Agreement". In: *DISC*. 2009.

[GHS12] C. Gentry, S. Halevi, and N. P. Smart. "Fully Homomorphic Encryption with Polylog Overhead". In: *EUROCRYPT*. 2012.

[GJPR21] A. Goel, A. Jain, M. Prabhakaran, and R. Raghunath. "On Communication Models and Best-Achievable Security in Two-Round MPC". In: *TCC*. 2021.

[GKOPZ20] J. Garay, A. Kiayias, R. M. Ostrovsky, G. Panagiotakos, and V. Zikas. "Resource-Restricted Cryptography: Revisiting MPC Bounds in the Proof-of-Work Era". In: *EUROCRYPT*. 2020.

[GOS06] J. Groth, R. Ostrovsky, and A. Sahai. "Perfect Non-interactive Zero Knowledge for NP". In: *EUROCRYPT*. 2006.

[Hye22] T. K. and Hyesun Kwak and Dongwon Lee and Jinyeong Seo and Yongsoo Song. *Asymptotically Faster Multi-Key Homomorphic Encryption from Homomorphic Gadget Decomposition*. Cryptology ePrint Archive, Report 2022/347. `https://ia.cr/2022/347`. 2022.

[JVC18] C. Juvekar, V. Vaikuntanathan, and A. Chandrakasan. "GAZELLE: A low latency framework for secure neural network inference". In: *USENIX Security Symposium*. 2018.

[Kia+21] A. Kiayias, C. Moore, S. Quader, and A. Russell. *Efficient Random Beacons with Adaptive Security for Ungrindable Blockchains*. IACR ePrint 2021/1698. `https://ia.cr/2021/1698`. 2021.

[KÖ22] J. Klemsa and M. Önen. *Parallel Operations over TFHE-Encrypted Multi-Digit Integers*. 2022.

[Lam06] L. Lamport. "Lower Bounds for Asynchronous Consensus". In: *Distrib. Comput.* (2006).

[Lam78] L. Lamport. "Time, Clocks, and the Ordering of Events in a Distributed System". In: *Commun. ACM* (1978).

[LJLA17] J. Liu, M. Juuti, Y. Lu, and N. Asokan. "Oblivious neural network predictions via minionn transformations". In: *CCS*. 2017.

[LM21] B. Li and D. Micciancio. "On the Security of Homomorphic Encryption on Approximate Numbers". In: *EUROCRYPT*. 2021.

[LPR13b] V. Lyubashevsky, C. Peikert, and O. Regev. "A toolkit for ring-LWE cryptography". In: *EUROCRYPT*. 2013.

[LSP82] L. Lamport, R. E. Shostak, and M. C. Pease. "The Byzantine Generals Problem". In: *ACM Trans. Program. Lang. Syst.* (1982).

[PS18] R. Pass and E. Shi. "Thunderella: Blockchains with Optimistic Instant Confirmation". In: *EUROCRYPT*. 2018.

[RU21] M. Rambaud and A. Urban. *Almost-Asynchronous MPC under Honest Majority, Revisited*. IACR ePrint 2021/503. 2021.

[SV14] N. P. Smart and F. Vercauteren. "Fully Homomorphic SIMD Operations". In: *Des. Codes Cryptography* (2014).