Collaborative Privacy-Preserving Analysis of Oncological Data using Multiparty Homomorphic Encryption

Ravit Geva[a], Alexander Gusev[b], Yuriy Polyakov[c], Lior Liram[c], Oded Rosolio[c], Andreea Alexandru[c], Nicholas Genise[c], Marcelo Blatt[c], Zohar Duchin[c], Barliz Waissengrin[a], Dan Mirelman[a], Felix Bukstein[a], Deborah T. Blumenthal[a], Ido Wolf[a], Sharon Pelles-Avraham[a], Tali Schaffer[a], Lee A. Lavi[a], Daniele Micciancio[c,d], Vinod Vaikuntanathan[c,e], Ahmad Al Badawi[c], and Shafi Goldwasser[c,f]

[a]Tel Aviv Sorasky Medical Center, Tel Aviv, Israel 64239
[b]Dana-Farber Cancer Institute, Harvard Medical School, Boston, MA 02215
[c]Duality Technologies, Inc, Hoboken, NJ 07103
[d]University of California, San Diego, CA 92093
[e]Massachusetts Institute of Technology, Cambridge, MA 02139
[f]Simons Institute for the Theory of Computing, University of California, Berkeley, CA 94720

August 8, 2023

Abstract

Real-world healthcare data sharing is instrumental in constructing broader-based and larger clinical data sets that may improve clinical decision-making research and outcomes. Stakeholders are frequently reluctant to share their data without guaranteed patient privacy, proper protection of their data sets, and control over the usage of their data. Fully homomorphic encryption (FHE) is a cryptographic capability that can address these issues by enabling computation on encrypted data without intermediate decryptions, so the analytics results are obtained without revealing the raw data. This work presents a toolset for collaborative privacy-preserving analysis of oncological data using multiparty FHE. Our toolset supports survival analysis, logistic regression training, and several common descriptive statistics. We demonstrate using oncological data sets that the toolset achieves high accuracy and practical performance, which scales well to larger data sets. As part of this work, we propose a novel cryptographic protocol for interactive bootstrapping in multiparty FHE, which is of independent interest. The toolset we develop is general-purpose and can be applied to other collaborative medical and healthcare application domains.

Contents

1 Introduction

There is a growing recognition of the important contribution of real-world data (RWD) in supporting healthcare decision-making in general [12, 24] and specifically in oncology [18, 30]. RWD are routinely collected from a variety of sources, such as electronic health records; medical claims and billing data; product and disease registries; and mobile devices [22]. RWD can complement data generated from randomized control trials (RCTs). While RCTs analyze data collected from controlled, limited, and homogenous patient populations, RWD allow the evaluation of larger and broader-based patient populations within the context of routine clinical practice [44]. Sharing RWD between several data owners results in a more complete data set than that obtained from a single data source and thus allows broader data analyses for better decision-making [1, 17]. In addition, healthcare data can be viewed as a revenue-producing asset that can be monetized. RWD analysis can save costs to the pharmaceutical industry by improving the identification of target populations, endpoints, and inclusion criteria, and thereby the overall study design [45]. Some of the main challenges of using RWD for healthcare decision-making are the facts that healthcare data are fragmented and originate from multiple sources, and that stakeholders are frequently hesitant to share or integrate their data, mainly due to trust issues [49].

Generally, patient data may be shared only if the patient's consent had been obtained for a given purpose or if the data are anonymized or deidentified [19]. While patients participating in RCTs can give their consent for data sharing, patients for whom RWD are collected do not necessarily provide their consent in advance for this purpose. In such cases data anonymization is required; however, anonymization procedures are recognized as being time-consuming, requiring manual intervention that can result in human error, difficult to scale, and challenging in terms of assessing their results [15]. Furthermore, anonymization requires the removal of sufficient patient data to prevent any possible re-deidentification, many times resulting in the impairment of scientific analysis and utility [37]. Healthcare data challenges, particularly patient privacy, data ownership, and data fragmentation, call for a data collaboration technology that, on the one hand, allows different parties to share their data and analytics, and, on the other hand, protects patient privacy and data ownership.

The two common cryptographic approaches to share and analyze sensitive data without compromising patient privacy and data ownership are secure MultiParty Computation (MPC) and Fully Homomorphic Encryption (FHE).[1] Both allow performing computations over encrypted data, but the underlying mechanisms are different. MPC, which was introduced by Yao in the 1980s [51], uses an approach where each party holds a secret and they perform computations on masked data using an interactive protocol. MPC is communication-bound and typically based on either garbled circuits or secret-sharing schemes [39]. FHE, which was first achieved by Gentry in 2008 [25], provides a non-interactive mechanism for performing computations on encrypted data in an untrusted environment, without ever decrypting the data or intermediate results. Only once the final computation results are obtained, the decryption of results may be performed by a different party that has the underlying secret key. FHE is compute-bound and typically based on lattice cryptography, which is resistant to attacks by quantum computers [13, 41].

Notable recent studies on privacy-preserving analysis of individual-level healthcare data using MPC include Cho et al. [16] and Hie et al. [32]. Cho et al. [16] report on large-scale genome-wide

[1]Our paper uses a number of specialized terms in cryptography and oncology; for convenience, we provide a glossary of these terms in Table A21 in the Appendices.

analysis of genotypic and phenotypic data using MPC. They perform a Genome-Wide Associate Study (GWAS) by dividing data among multiple servers and computing the GWAS via MPC among the servers. They demonstrate that their results provide adequate accuracy, and reasonable runtime (about 37 hours) can be achieved for problem sizes of 100,000 individuals and 500,000 single nucleotide polymorphisms, enabling real-scale privacy-preserving GWAS. Hie et al. [32] develop a computational protocol for securely training a predictive model of drug–target interactions on a pooled data set using MPC. Their protocol for neural network training runs within days on a real data set of more than one million interactions and is more accurate than state-of-the-art drug-target interaction prediction methods.

FHE has also seen significant success in performing privacy-preserving analysis for certain healthcare use cases. Note that almost all FHE results described below are based on the Cheon-Kim-Kim-Song (CKKS) FHE scheme [14], which is the most efficient scheme for real-number arithmetic and many machine learning applications [41]. For instance, Blatt et al. [7] demonstrate that FHE can perform GWAS for 100,000 individuals and 500,000 single nucleotide polymorphisms in less than 6 hours, hence achieving a better runtime than the prior MPC approach of Cho et al. [16] while still providing a comparable accuracy. Kim et al. [36] were able to train a logistic regression model using an encrypted data set for 1,579 individuals with 18 binary genotypes and a binary phenotype outcome (cancer/no cancer). Using several aggressive approximations and optimized values of tunable parameters, the authors were able to perform encrypted logistic regression training in about 6 minutes on a commodity desktop machine.

However, practical results with FHE can typically be achieved only for relatively *shallow* (limited-depth) computations that do not require bootstrapping, a special procedure that refreshes *exhausted* ciphertexts to enable more computations. Bootstrapping is a computationally expensive and memory-intensive procedure that needs to be invoked many times for *deep* computations such as logistic regression training or deep neural network inference. In applications with bootstrapping, the FHE runtimes and memory consumption become much higher. Notable recent studies implementing machine learning capabilities using CKKS bootstrapping are Han et al. [31] and Lee et al. [38]. Han et al. [31] present a logistic regression training capability based on FHE that can train a model with 422,108 samples over 200 features in about 17 hours. Lee et al. [38] develop a privacy-preserving CNN inference solution that can classify with a ResNet-56 model a CIFAR-10 image in about 2 hours. In both cases, most of the computation time is spent on CKKS bootstrapping.

To minimize the number of CKKS bootstrapping invocations in applications of FHE, researchers often use hand-tuned low-accuracy–low-degree approximations for nonlinear functions and data-set-optimized parameters, e.g., learning rate, which allows to significantly improve the efficiency of an FHE computation for a given data set. But as soon as the FHE solution is applied to other data sets, the solution stops working correctly or achieving an adequate accuracy. For example, Han et al. [31] used a degree-3 polynomial approximation of the sigmoid function obtained using the least squares method for the range of $[-8, 8]$. Our analysis of polynomial sigmoid approximations in the Nesterov gradient descent method of logistic regression training (same method as in [31]) for another large data set shows that a Chebyshev interpolation in the range of $[-32, 32]$ using a polynomial of degree of at least 32 is needed to achieve satisfactory accuracy results (see Appendices for details). Generally speaking, both the range and polynomial degree may significantly vary from one data set to another. If a more costly polynomial approximation is used, the bootstrapping has to be invoked much more frequently. For comparison, the logistic regression solution in [31] performed bootstrapping every 5 iterations whereas ours calls bootstrapping after each iteration.

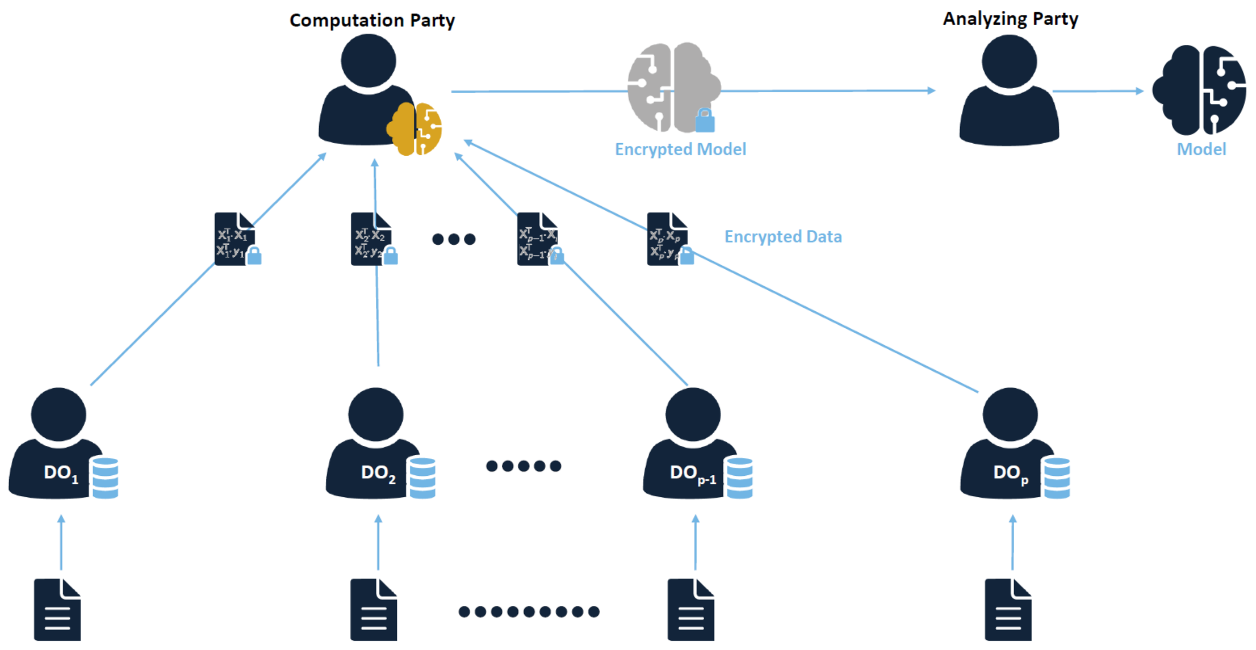


Figure 1: Schematic of multiparty (threshold) FHE. Any party may have a secret share (assignment of secret shares is determined by the use case). At least two parties have secret shares. First, all parties with secret shares perform distributed key generation to compute the common public key, corresponding to the sum of secret shares. Next, the data are encrypted by each Data Owner (DO) using the common public key. Then, the computation is performed by the Computation Party (CP). If interactive bootstrapping is needed, the CP interacts with the parties that have secret shares. Finally, the encrypted result is decrypted using a distributed decryption procedure involving all parties with secret shares. The Analyzing Party (AP) is the party that gets to see the result of the computation, and can be the same as one of DOs (multiple DOs may serve as APs in some use cases). In the setting of multiparty FHE, the CP can be one of the DOs. The DOs, CP, and AP are separated in the schematic to show all possible roles involved in the multiparty FHE collaboration model.

To address the FHE bootstrapping inefficiency, Froelicher et al. [23] present an interactive computation framework based on multiparty FHE (the algorithms were originally introduced in [43]), which uses FHE for most of the computations and interactive techniques for bootstrapping and several other operations. In multiparty FHE (typically referred to as *threshold FHE* in cryptography literature [3]), each party may have a secret share (similar to classical MPC based on secret sharing), and distributed key generation and decryption protocols are executed involving all parties with secret shares (see Figure 1). The main efficiency benefit of this approach as compared to FHE is that bootstrapping can be done interactively much faster (by two orders of magnitude or even more) than in the classical FHE setting. The authors demonstrate the use of their privacy-preserving framework for Kaplan-Meier survival analysis in oncology and genome-wide association studies in medical genetics. Froelicher et al. [23] consider the *federated* collaboration model between data owners, where each party contributes a subset of records to the full data set used for privacy-preserving analysis (see Figure 2a).

Our work extends and improves the multiparty FHE framework of [23] in several different ways.

First, we add the *private join* collaboration model where multiple parties can contribute data for the same records (e.g., individuals) in a way where the data owners do not learn which records match (with only the computation party learning the intersection size in the case of two data owners), and this joined data is then used for further analysis using multiparty FHE (see Figure 2b).

Second, we introduce a novel, more efficient interactive bootstrapping procedure for the case of two parties and improve the more general (for any number of parties) interactive bootstrapping method used in [23], and initially proposed in [43].

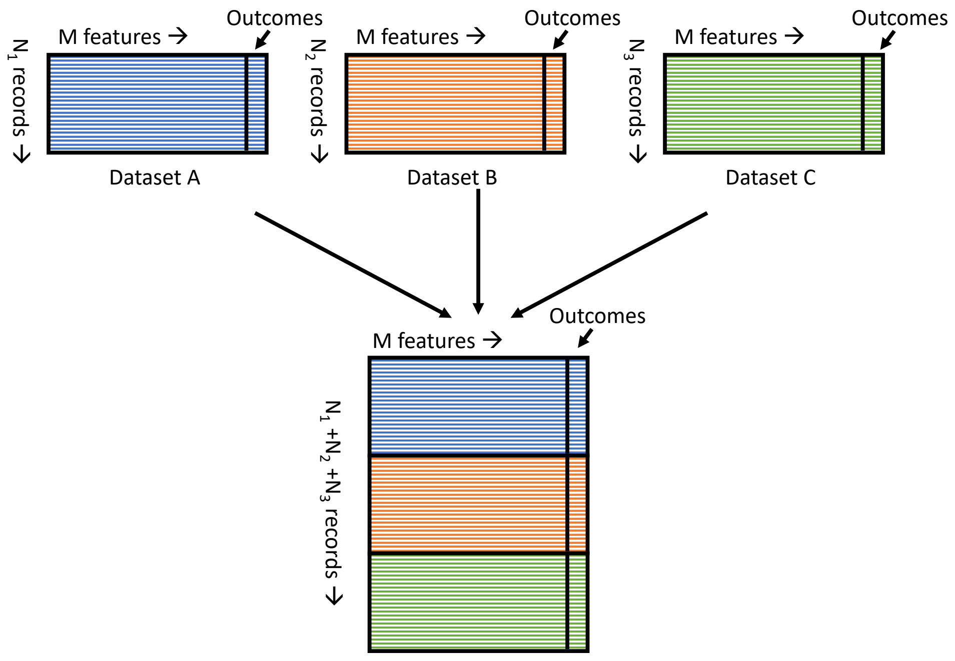
Third, we extend the list of computations to provide a more general toolset for the privacy-preserving analysis of oncological data. The computations implemented in our work include mean, median, standard deviation, frequency, χ^2 test, t test, survival analysis (Kaplan-Meier plots and log-rank test), and logistic regression training over encrypted data.

2 Results

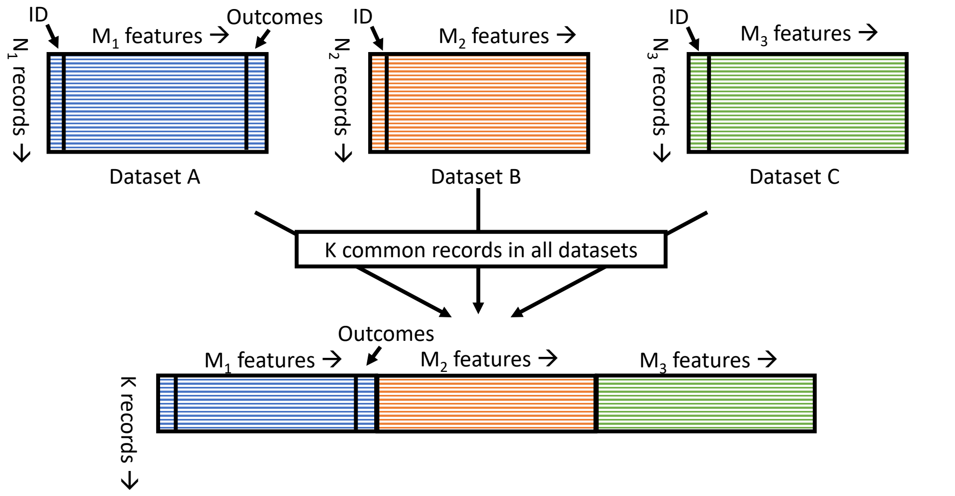
We applied our multiparty FHE toolset to two different oncological data sets: a real-world data set of colorectal cancer patients' survival data at the Tel Aviv Sourasky Medical Center and a previously published data set based on two clinical trials of immunotherapy in renal cell carcinoma [9].

The real-world data set of colorectal cancer patients' survival data includes 623 patients and 24 variables, amounting to 14,952 items of data. The goal of the study was to examine the effect of oxaliplatin treatment with and without cannabis for patients with colorectal cancer. Statistical analysis of key oncological endpoints was blindly performed on both the raw data and FHE-encrypted data using descriptive statistics and survival analysis with Kaplan-Meier curves and log-rank tests. The results were then compared with an accuracy goal of two decimals. Early results of this study (for the single-key FHE setting) are reported in [27]. The study included the following statistical analyses: mean, median, and standard deviation for the age of cancer onset; frequency analysis for sex; χ^2-test between cannabis indicator (with or without cannabis) and diagnosis, χ^2-test between cannabis indicator and sex; t-test for cannabis indicator by age of onset. Kaplan-Meier and log-rank survival analysis was performed to examine the effect of the treatment with cannabis on the overall survival of patients.

All accuracy metrics were found to be within the pre-determined accuracy goal of two decimal digits. The Kaplan-Meier curves for both the data in the clear and encrypted data are illustrated in

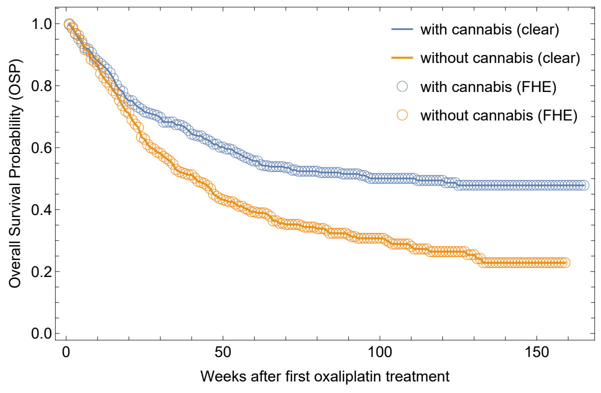


(a) Federated model: each party contributes a subset of records to the full data set used for privacy-preserving analysis. This model supports two scenarios: (1) local FHE computations are performed by each DO (similar to the federated learning setting) and (2) an FHE computation is carried out by CP on the stacked encrypted data set.



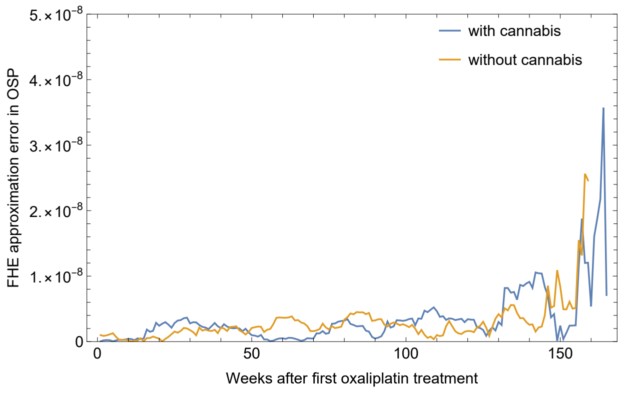
(b) Private join model: multiple parties contribute features data for the same records in a way where the parties do not learn which records match. Then an FHE computation is performed on the linked encrypted data.

Figure 2: Collaboration models for privacy-preserving analysis of data from multiple DOs.



(a) Kaplan-Meier curves: in the clear vs FHE. The number of patients at risk:

Time (weeks)	25	75	125	175
With cannabis	239	138	78	58
Without cannabis	180	90	32	18



(b) FHE approximation error computed as the absolute value of the difference in Overall Survival Probability (OSP) between the results in the clear and FHE results. The error in OSP gradually increases with time as expected, since at each time step the number of (FHE) computations for that value increases as well. However, the error is less than 5×10^{-8}, which is negligible for the survival analysis.

Figure 3: Kaplan-Meier survival analysis of the real-world data set for colorectal cancer patients: results in the clear vs encrypted data (a), FHE approximation error (b). There were two groups of patients treated with oxaliplatin: first group was taking cannabis and the other was not. The survival analysis results for the encrypted dataset were found to be accurate up to 7 decimal digits compared to the results for the unencrypted data set (see also Table A1 in the Appendices for numerical details). Note that this Kaplan-Meier analysis has no clinical significance and should not be interpreted as such. The analysis was performed solely for the purpose of testing the proposed FHE method.

Figure 3. The numerical results of the first 15 weeks out of 141 weeks following the first oxaliplatin treatment are listed in Table A1 in the Appendices. The runtime of less than half a minute was observed for descriptive statistics and about three minutes for the survival analysis [27]. Note that the time of the anonymization and statistical analyses performed on the raw data set by a statistician, the method commonly used in clinical oncology, is estimated to be about 10 hours, which is significantly higher than the runtime of FHE computations. As this data set is not publicly available (see the Data Availability section for more details), we performed a similar analysis for a publicly available data set so that our results could be independently reproduced. We further extended the analysis to include logistic regression training, another useful tool for oncological and broader healthcare studies.

Next, we show an example of applying our multiparty FHE toolset to an analysis of a previously published data set, providing detailed results for it in the Appendices. Individual-level data from two clinical trials of immunotherapy in renal cell carcinoma were accessed from prior publications [9]. In brief, a PD-1 immune checkpoint inhibitor (nivolumab) was evaluated for 1,006 patients with advanced clear-cell renal cell carcinoma (ccRCC) in the CheckMate 025 and CheckMate 010 randomized clinical trials, as compared to the standard of care with mTOR inhibitors (everolimus). Clinical outcomes data included overall survival and progression-free survival as well as basic patient demographics, and genomic data included tumor whole exome sequencing. Prior work had identified a survival benefit for nivolumab as well as improved progression-free survival for the subset of patients with mutations in PBRM1, and we focused on these positive controls in our analyses here.

First, we evaluated the accuracy of basic demographic summaries of age, sex, prior treatment, and objective response rate (ORR) within and across the treatments. More concretely, we computed the mean, median, and standard deviation for age; performed a χ^2-test between ORR and trial arm, where trial arm was set to 1 for nivolumab and 0 for everolimus; performed t-tests for age by trial arm (t-test 1) and age by ORR groups (t-test 2); we evaluated the frequency for sex (frequency 1), benefit (frequency 2), PBRM1 (frequency 3), and the number of prior therapies (frequency 4).

Second, we conducted survival analyses where mortality was the end-point and patients were censored at loss-to-follow-up, with statistical significance assessed by log-rank test and Kaplan-Meier analysis. For the treatment arm positive control (nivolumab vs other), which corresponds to Kaplan-Meier and log-rank scenario 1, we observed a significant association, e.g., the p-value for log-rank test was 0.001. For the sex-stratified negative control, which corresponds to Kaplan-Meier and log-rank scenario 2, as expected, we observed no significant difference between groups, e.g., the p-value for the log-rank test was 0.104.

Third, we conducted biomarker survival analyses where progression-free survival was the end-point and patients were censored at loss-to-follow-up. For the positive control within the nivolumab arm, which corresponds to Kaplan-Meier and log-rank scenario 3, patients with PBRM1 mutations exhibited significantly longer survival than non-carriers by long-rank test, e.g., the p-value for the log-rank test was 0.006.

Fourth, we conducted a logistic regression analysis where ORR was the outcome and age, sex, and trial arm where independent variables. For this analysis, ORR was defined as 1 for Complete Response or Partial Response (CR/PR) and 0 for Stable Disease or Progression Disease (SD/PD). As expected, a significant association was observed with trial arm.

For the multiparty FHE experiments, the full data set was filtered down and broken into different subsets to emulate realistic private join scenarios with two data owners (see Table A2 in the Appendices for details). Note that the runtime and communication costs for the private join protocol are negligibly small as the numbers of records and features for the oncological data set are not high. Hence, the runtimes reported here are determined by the FHE computations performed after executing the join protocol.

Table 1 shows the relative errors for descriptive statistics and survival analysis, as compared to the results in the clear. For all computations, accuracy of more than 5 decimal digits (as compared to the computations in the clear) was achieved. Note that for frequency computations, the error was zero because we used BFV, an exact homomorphic encryption scheme, for these computations. Table 2 illustrates the runtime and storage performance. Besides more complex survival analysis, all computations take less than half a minute. The survival analysis takes up to one minute and a half. The memory requirements do not exceed a few gigabytes. These results imply that privacy-preserving descriptive statistics and survival analysis using multiparty FHE are already practical for typical oncological data sets.

Our results for logistic regression training using multiparty FHE suggest that accuracy of at least 6 decimal digits was achieved after 100 iterations (see the Appendices). The performance results for logistic regression training are illustrated in Table 3. One iteration takes about 5 seconds and the overall runtime of 500 seconds is observed for 100 iterations. The memory requirements do not

Table 1: Numerical accuracy for descriptive statistics and survival analysis computed with FHE vs the computations in the clear using the data published in [9]. The mean, standard deviation, and median were computed for age; *t*-test 1 for age by trial arm; *t*-test 2 for age by ORR groups; χ^2-test for ORR by trial arm; Kaplan-Meier and log-rank 1 for overall survivability by arm (OS, OS_CNRS); Kaplan-Meier and log-rank 2 for overall survivability by sex (OS, OS_CNRS); Kaplan-Meier and log-rank 3 for progression-free survival with somatic mutations by arm (PFS, PFS_CNRS); frequency 1 for sex; frequency 2 for benefit; frequency 3 for PBRM1; frequency 4 for the number of prior therapies. All these computations did not require bootstrapping.

FHE Scheme	**Computation**	**Statistic**	**Rel Error**
CKKS	mean	average	2.83e-12
	standard deviation	std dev	1.62e-07
	median	quantile	0.0e+00
	t-test 1	t-score	1.57e-09
	t-test 2	t-score	1.60e-09
	χ^2	χ^2	1.91e-09
	Kaplan-Meier 1	probability	2.04e-07
	Kaplan-Meier 2	probability	7.39e-06
	Kaplan-Meier 3	probability	2.08e-07
	log-rank 1	χ^2	3.21e-08
	log-rank 2	χ^2	4.92e-08
	log-rank 3	χ^2	3.80e-08
BFV	frequency 1	count	0.0e+00
	frequency 2	count	0.0e+00
	frequency 3	count	0.0e+00
	frequency 4	count	0.0e+00

exceed 9 gigabytes. Both the runtime and memory are significantly smaller than for the scenario of non-interactive FHE bootstrapping, which makes it possible to run training on a commodity desktop or server machine. For instance, one non-interactive CKKS bootstrapping operation for this setting takes at least 24 seconds [5].

To evaluate the scalability of our multiparty FHE framework, we ran the most computationally intensive capability considered in our work, logistic regression training in the private join collaboration setting, for much larger problem sizes than the oncological data set. We considered the case of two DOs, where the first DO has the encrypted outcomes data and the second DO has the features data. In this case, only encrypted outcomes data need to be sent to the party that sees the result of the private join, which implies that the performance cost of private join is almost negligible as compared to logistic regression training. We ran performance experiments on a server commodity system with 72 threads for simulated data sets from 16,384 samples to 1,048,576 samples (doubling each time), all with 256 features. For the sample sizes from 16,384 to 262,144, the runtime of one iteration stayed roughly the same at about 60 seconds. For 524,288 and 1,048,576 samples, the runtime was 98 and 187 seconds, respectively. This implies that our logistic regression solution scales relatively well with the number of cores of the server system; more concretely, the runtime degrades only by a factor of 3 when increasing the number of samples by a factor of 64.

Table 2: Runtime and storage measurements for computations in Table 1; Kaplan-Meier and log-rank survival analysis methods are abbreviated as KM and LR, respectively; Keygen and Comp correspond to key generation and computation.

	Peak RAM	**Keygen**	**Comp**	**Keygen**	**Comp**
	`[GB]`	`[sec]`	`[sec]`	`[MB]`	`[MB]`
mean	2.13	1.2	1.6	140	1.0
standard deviation	2.51	6.2	2.5	600	3.0
median	2.17	2.0	21.7	244	108.0
t-test 1	1.97	2.9	3.2	477	1.3
t-test 2	1.99	2.9	3.1	477	1.3
χ^2	2.43	9.7	6.7	1,458	1,458.0
KM & LR 1	2.36	3.1	79.3	546	3,030.0
KM & LR 2	2.39	2.7	79.6	546	3,030.0
KM & LR 3	2.43	2.9	20.7	546	727.0
frequency 1	2.03	0.3	1.2	23	0.5
frequency 2	2.06	0.3	1.2	23	0.8
frequency 3	2.03	0.3	1.2	23	0.8
frequency 4	2.13	0.3	1.2	23	1.3

Table 3: Runtime and memory performance for logistic regression training; interactive bootstrapping is performed after every iteration; KeyGen refers to key generation.

Iterations	**Peak RAM**	**Total**	**Keygen**	**Iteration time**
	`[GB]`	`[sec]`	`[sec]`	`[sec]`
10	7.752	83.4	30.6	5.3
20	7.943	128.3	30.7	4.9
30	8.072	175.0	30.5	4.8
40	8.199	222.5	30.8	4.8
50	8.257	264.9	30.7	4.7
60	8.273	314.7	31.2	4.7
70	8.630	362.4	30.6	4.7
80	8.625	414.2	31.4	4.8
90	8.703	450.0	30.7	4.7
100	8.760	500.0	30.7	4.7

3 Discussion

Our approach enables multiple new analyses of clinical and genetic data.

First, the federated model allows multiple institutions with disjoint clinical and genomic data to perform secure joint analyses across all patients without decrypting the underlying individual-level values. Clinical trials conducted across multiple institutions can now evaluate drug efficacy by computing secure Kaplan-Meier and log-rank analyses. In particular, genomic data, which is often considered sensitive patient data that may be impossible to de-identify, can be leveraged to perform biomarker analyses within or across treatment subgroups to identify treatment modifiers. We demonstrate the feasibility and accuracy of this approach by recapitulating the protective effect of somatic PBRM1 mutations in patients on immune checkpoint inhibitors [42]. Such genetic biomarkers can prioritize effective treatments for patients who would not have otherwise received them and further improve patient outcomes as well as expand our understanding of disease etiology. In addition to analyses of survival-related outcomes, recent studies have identified novel biomarkers associated with drug safety outcomes through time-to-event analyses of adverse events [29]. While overall adverse event rates are reported with clinical trials, individual-level data is typically not made available. Our methodology would enable secure, large-scale adverse event studies across multiple trials with the potential of identifying actionable predictors of toxicities that can be mitigated before they occur.

Second, the private join model allows investigators to conduct secure analyses of clinical and genomic data when the underlying data reside at different sites and cannot be integrated in the clear. This is often the case for biobank cohorts, which routinely collect rich genetic/genomic data but typically only sparse clinical measurements through billing codes; whereas detailed chart review and clinical records abstraction is typically conducted under strict Institutional Review Board protocols at medical institutions but which do not conduct routine biobanking. For example, while the UK Biobank has collected a wide array of omics data including whole-exome and whole-genome sequencing, it only has rudimentary treatment billing codes and does not report the duration, dosage, or treatment response [50]. Individual institutions that collect such data but do not conduct genotyping could thus use our approach to tap into existing genomic resources while retaining patient confidentiality across both cohorts. While genetic data was used for demonstration in this study, our approaches would naturally apply to broader classes of omics that can be summarized as continuous or categorical values, such as magnetic resonance images [47] or structural brain images [20]. Our approach thus enables privacy-preserving identification of clinical biomarkers across institutions and data silos.

Our approach has several limitations and areas for future work. First, our private join approach requires sending the full encrypted DO data sets (the fields to be computed on) to the party computing the join of the data sets from DOs. This requirement can only be removed if the DOs are allowed to learn something about the intersection. Second, the extension of private join to more than two DOs prevents DOs from learning anything about the intersection only if the computing party performs expensive homomorphic re-arrangement of encrypted data (using many rotations) or if the DOs are not allowed to collude with each other. Devising a more efficient solution for extending the private join to more than two DOs is left for future work. Third, while the FHE scheme we use is plausibly post-quantum secure, the private join protocol in our framework uses a commutative deterministic cipher based on an elliptic-curve instantiation of the Decisional Diffie-Hellman (DDH) problem, which is secure against classical computer attacks but is not quantum-resistant. As future work, we will look into developing a quantum-resistant version of this private join protocol based on lattices. See the Appendices for more details on the first three limitations. Fourth, our multiparty FHE framework is based on the same adversarial model as single-key FHE, i.e., it is secure against semi-honest adversaries.

4 Materials and Methods

Software Implementation

We implemented our multiparty FHE framework in PALISADE v1.11.9 [46], an open-source lattice cryptography software library that includes all common FHE schemes. For the experiments, we used full Residue Number System (RNS) variants of the Cheon-Kim-Kim-Song (CKKS) and Brakerski/Fan-Vercauteren (BFV) FHE schemes [8, 14, 21], which are already available in PALISADE. The full RNS variants of CKKS and BFV are described in [34] and [35], respectively.

Multiparty FHE

Our implementation is based on the threshold FHE construction proposed in [3]. We consider the scenario of additive secret sharing where the sum of all secret shares corresponds to the underlying secret key, but this secret key is never revealed. PALISADE provides threshold FHE extensions (without interactive bootstrapping) for the CKKS and BFV schemes [46].

Private Join

The private join collaboration model is a generalization of the private intersection-sum-with-cardinality protocol proposed in [33]. We extend the original protocol from encrypted summation to arbitrary encrypted computations, and add support for two or more DOs with FHE-encrypted data. The model includes a CP and multiple DOs. Each DO has a subset of features for common records (all DO data sets are encrypted using FHE). The purpose of join is to link the data sets from DOs into a single data set and perform FHE computations on it. The same record identification scheme is used for all DOs, i.e., each common record is uniquely described by the same identifier. The join is performed based on exact matches. The CP computes the intersection by interacting with the DOs, and the DOs also interact with each other. A deterministic commutative cipher based on an elliptic curve instantiation of the DDH problem is used to compute a common hash (encryption) for each record. As part of the protocol, the records get randomly shuffled and random identifiers get inserted. In the case of two DOs, the CP learns the intersection size whereas the DOs learn nothing about the intersection. The protocol and security proofs for it are described in detail in the Appendices.

Interactive Bootstrapping

Our multiparty FHE framework includes two interactive bootstrapping procedures. The first algorithm achieves more efficient bootstrapping for two parties, and is a novel contribution of our work. In contrast to the more general procedure of [43] that requires at least three extra RNS residues, our algorithm requires only one extra RNS residue, which reduces the computational complexity of the full FHE workflow (not only the interactive bootstrapping invocations). Our procedure is based on a technique of distributed rounding. We describe the CKKS instantiation of our 2-party interactive bootstrapping protocol along with the security proofs in the Appendices. The second algorithm supports any number of parties and is an optimized version of the procedure proposed in [43]. Our variant reduces the computational complexity associated with sampling: one polynomial Gaussian sampling is eliminated because it is not needed for security. We implement both protocols in PALISADE.

Experimental Test Bed

FHE computations for the oncological data set were performed using a server with Intel(R) Xeon(R) Platinum 8275CL @ 3.00GHz and 96 GB of RAM. The experiments were run at 16 threads using the PALISADE multithreading capability based on OMP. The server had Ubuntu 20.04 and g++ (GCC) 9.3.0 installed. All parties were connected via a local area network connection. Note that interactive computations were needed only for the logistic regression training case, where the communication requirements for interactive bootstrapping are small (a single ciphertext is transferred between the interacting parties). The scalability experiments were run on a Ubuntu 20.04 server with Intel(R) Xeon(R) Platinum 8360Y CPU @ 2.40GHz (72 threads) and 128 GB RAM.

Computations

Before running the FHE computations, the source data was pre-processed to normalize the input data and exclude NA records. Then, analysis in the clear was performed to generate reference results for evaluating the accuracy of FHE analysis. The FHE computations for mean, standard deviation, median, χ^2-test, t-test, survival analysis (Kaplan-Meier plot and log-rank test), and logistic regression training were performed using the CKKS scheme. The CKKS in PALISADE was configured to use hybrid key switching, the scaling factor size was set to 50 bits (55 for standard deviation), the standard mode without automatic rescaling was used, the ring dimension was automatically set to the minimum value required for achieving 128 bits of security, and the multiplicative depth was set to the minimum needed to achieve the correctness. All other parameters were set to PALISADE defaults for CKKS. For frequency computations, we used the BFV scheme configured to use the Brakerski-Vaikuntanathan key switching and plaintext modulus of 1032193. The ring dimension was automatically set to the minimum value required for achieving 128 bits of security and the multiplicative depth was set to the minimum needed to achieve correctness. All other parameters were set to PALISADE defaults for BFV. For logistic regression training, we used the 2-party interactive bootstrapping proposed in this work. For all computations, we used a 2-DO private join setup, i.e., the full data set was filtered down and broken into different subsets to emulate realistic private join scenarios with two DOs. More details about the computations and private join setup are provided in the Appendices.

Data Availability

The colorectal cancer patients's data sets generated and/or analyzed during the current study are not publicly available due to patient's privacy. Personal patient information was anonymized and stored on a password-protected computer. The computer is located in a locked office of the investigator. The data that support the findings of this study are available from Dr. Ravit Geva but restrictions apply to the availability of these data, which were used under license for the current study, and so are not publicly available. The data are, however, available from the authors upon reasonable request and with permission of the Tel Aviv Sourasky Medical Center Helsinki Committee.

Our main experiments used a previously published data set based on two clinical trials of immunotherapy in renal cell carcinoma [9].

Code Availability

The PALISADE version we used is publicly available in GitLab at `https://gitlab.com/palisade/palisade-release`https://gitlab.com/palisade/palisade-release. The code with the FHE computations, including descriptive statistics, survival analysis, and logistical regression training, is currently not publicly available as its license does not allow for open-source redistribution. However, the pseudocode of all new algorithms introduced in our work, namely, private join and interactive bootstrapping, is provided in the Appendices. Upon request sent to the corresponding author(s), we can provide binaries that, in combination with open-source resources, can be used for the sole purpose of verifying and reproducing the experiments in the manuscript.

References

[1] S. Adimadhyam, E. F. Barreto, N. M. Cocoros, S. Toh, J. S. Brown, J. C. Maro, J. Corrigan-Curay, G. J. Dal Pan, R. Ball, D. Martin, M. Nguyen, R. Platt, and X. Li. Leveraging the Capabilities of the FDA's Sentinel System To Improve Kidney Care. *J Am Soc Nephrol*, 31(11):2506–2516, Nov 2020.

[2] M. R. Albrecht, M. Chase, H. Chen, J. Ding, S. Goldwasser, S. Gorbunov, S. Halevi, J. Hoffstein, K. Laine, K. E. Lauter, S. Lokam, D. Micciancio, D. Moody, T. Morrison, A. Sahai, and V. Vaikuntanathan. Homomorphic encryption standard. *IACR Cryptol. ePrint Arch.*, page 939, 2019.

[3] 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 D. Pointcheval and T. Johansson, editors, *Advances in Cryptology – EUROCRYPT 2012*, pages 483–501, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[4] G. Asharov, A. Jain, and D. Wichs. Multiparty computation with low communication, computation and interaction via threshold FHE. *IACR Cryptol. ePrint Arch.*, page 613, 2011.

[5] A. A. Badawi and Y. Polyakov. Demystifying bootstrapping in fully homomorphic encryption. Cryptology ePrint Archive, Paper 2023/149, 2023. `https://eprint.iacr.org/2023/149`.

[6] D. J. Bernstein. Curve25519: New diffie-hellman speed records. In *Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 207–228. Springer, 2006.

[7] M. Blatt, A. Gusev, Y. Polyakov, and S. Goldwasser. Secure large-scale genome-wide association studies using homomorphic encryption. *Proc Natl Acad Sci U S A*, 117(21):11608–11613, May 2020.

[8] Z. Brakerski. Fully homomorphic encryption without modulus switching from classical gapsvp. In *Advances in Cryptology–CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 868–886. Springer, 2012.

[9] D. A. Braun, Y. Hou, Z. Bakouny, M. Ficial, M. Sant' Angelo, J. Forman, P. Ross-Macdonald, A. C. Berger, O. A. Jegede, L. Elagina, J. Steinharter, M. Sun, M. Wind-Rotolo, J. C. Pignon, A. D. Cherniack, L. Lichtenstein, D. Neuberg, P. Catalano, G. J. Freeman, A. H. Sharpe, D. F. McDermott, E. M. Van Allen, S. Signoretti, C. J. Wu, S. A. Shukla, and T. K. Choueiri. Interplay of somatic alterations and immune infiltration modulates response to PD-1 blockade in advanced clear cell renal cell carcinoma. *Nat Med*, 26(6):909–918, Jun 2020.

[10] E. Bresson, Y. Lakhnech, L. Mazaré, and B. Warinschi. A generalization of DDH with applications to protocol analysis and computational soundness. In *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 482–499. Springer, 2007.

[11] D. Böhning and B. Lindsay. Monotonicity of quadratic-approximation algorithms. *Annals of the Institute of Statistical Mathematics*, 40(4):641–663, 1988.

[12] A. Cave, X. Kurz, and P. Arlett. Real-world data for regulatory decision making: Challenges and possible solutions for Europe. *Clinical Pharmacology & Therapeutics*, 106(1):36–39, 2019.

[13] J. H. Cheon, A. Costache, R. C. Moreno, W. Dai, N. Gama, M. Georgieva, S. Halevi, M. Kim, S. Kim, K. Laine, Y. Polyakov, and Y. Song. *Introduction to Homomorphic Encryption and Schemes*, pages 3–28. Springer International Publishing, Cham, 2021.

[14] J. H. Cheon, A. Kim, M. Kim, and Y. Song. Homomorphic encryption for arithmetic of approximate numbers. In T. Takagi and T. Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 409–437, Cham, 2017. Springer International Publishing.

[15] R. Chevrier, V. Foufi, C. Gaudet-Blavignac, A. Robert, and C. Lovis. Use and understanding of anonymization and de-identification in the biomedical literature: Scoping review. *J Med Internet Res*, 21(5):e13484, May 2019.

[16] H. Cho, D. J. Wu, and B. Berger. Secure genome-wide association analysis using multiparty computation. *Nat Biotechnol*, 36(6):547–551, Jul 2018.

[17] L. H. Curtis, J. Brown, and R. Platt. Four health data networks illustrate the potential for a shared national multipurpose big-data network. *Health Aff (Millwood)*, 33(7):1178–1186, Jul 2014.

[18] G. D. Demetri and S. Stacchiotti. Contributions of real-world evidence and real-world data to decision-making in the management of soft tissue sarcomas. *Oncology*, 99 Suppl 1:3–7, 2021.

[19] K. El Emam, S. Rodgers, and B. Malin. Anonymising and sharing individual patient data. *BMJ*, 350:h1139, Mar 2015.

[20] L. T. Elliott, K. Sharp, F. Alfaro-Almagro, S. Shi, K. L. Miller, G. Douaud, J. Marchini, and S. M. Smith. Genome-wide association studies of brain imaging phenotypes in UK Biobank. *Nature*, 562(7726):210–216, Oct 2018.

[21] J. Fan and F. Vercauteren. Somewhat practical fully homomorphic encryption. Cryptology ePrint Archive, Paper 2012/144, 2012. `https://eprint.iacr.org/2012/144`.

[22] Submitting documents using real-world data and real-world evidence to FDA for drug and biological products. `https://www.fda.gov/regulatory-information/search-fda-guidance-documents/submitting-documents-using-real-world-data-and-real-world-evidence-fda-drug-and-biological-products`, 2022. Accessed: 2023-01-20.

[23] D. Froelicher, J. R. Troncoso-Pastoriza, J. L. Raisaro, M. A. Cuendet, J. S. Sousa, H. Cho, B. Berger, J. Fellay, and J. P. Hubaux. Truly privacy-preserving federated analytics for precision medicine with multiparty homomorphic encryption. *Nat Commun*, 12(1):5910, Oct 2021.

[24] L. P. Garrison Jr., P. J. Neumann, P. Erickson, D. Marshall, and C. D. Mullins. Using real-world data for coverage and payment decisions: The ISPOR real-world data task force report. *Value in Health*, 10(5):326–335, 2007.

[25] C. Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing*, STOC '09, page 169–178, New York, NY, USA, 2009. Association for Computing Machinery.

[26] C. Gentry, S. Halevi, and N. P. Smart. Homomorphic evaluation of the AES circuit. In *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867. Springer, 2012.

[27] R. Geva, B. Waissengrin, D. Mirelman, F. Bokstein, D. T. Blumenthal, I. Wolf, S. Pelles, Z. Duchin, L. Liram, Y. Polyakov, and M. Blatt. Verification of statistical oncological endpoints on encrypted data: Confirming the feasibility of real-world data sharing without the need to reveal protected patient information. *Journal of Clinical Oncology*, 39(15_suppl):e18725–e18725, 2021.

[28] O. Goldreich. *The Foundations of Cryptography - Volume 2: Basic Applications*. Cambridge University Press, 2004.

[29] S. Groha, S. A. Alaiwi, W. Xu, V. Naranbhai, A. H. Nassar, Z. Bakouny, T. El Zarif, R. M. Saliby, G. Wan, A. Rajeh, E. Adib, P. V. Nuzzo, A. L. Schmidt, C. Labaki, B. Ricciuti, J. V. Alessi, D. A. Braun, S. A. Shukla, T. E. Keenan, E. Van Allen, M. M. Awad, M. Manos, O. Rahma, L. Zubiri, A. C. Villani, B. Fairfax, C. Hammer, Z. Khan, K. Reynolds, Y. Semenov, D. Schrag, K. L. Kehl, M. L. Freedman, T. K. Choueiri, and A. Gusev. Germline variants associated with toxicity to immune checkpoint blockade. *Nat Med*, 28(12):2584–2591, Dec 2022.

[30] J. P. Hall, J. Chang, R. Moon, O. Higson, K. Byrne, J. P. Doherty, and J. C. Cappelleri. Real-world treatment patterns in patients with advanced (stage III-IV) ovarian cancer in the USA and Europe. *Future Oncol*, 16(15):1013–1030, May 2020.

[31] K. Han, S. Hong, J. H. Cheon, and D. Park. Logistic regression on homomorphic encrypted data at scale. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):9466–9471, Jul. 2019.

[32] B. Hie, H. Cho, and B. Berger. Realizing private and practical pharmacological collaboration. *Science*, 362(6412):347–350, Oct 2018.

[33] M. Ion, B. Kreuter, A. E. Nergiz, S. Patel, S. Saxena, K. Seth, M. Raykova, D. Shanahan, and M. Yung. On deploying secure computing: Private intersection-sum-with-cardinality. In *IEEE European Symposium on Security and Privacy, EuroS&P 2020, Genoa, Italy, September 7-11, 2020*, pages 370–389. IEEE, 2020.

[34] A. Kim, A. Papadimitriou, and Y. Polyakov. Approximate homomorphic encryption with reduced approximation error. In S. D. Galbraith, editor, *Topics in Cryptology – CT-RSA 2022*, pages 120–144, Cham, 2022. Springer International Publishing.

[35] A. Kim, Y. Polyakov, and V. Zucca. Revisiting homomorphic encryption schemes for finite fields. In M. Tibouchi and H. Wang, editors, *Advances in Cryptology – ASIACRYPT 2021*, pages 608–639, Cham, 2021. Springer International Publishing.

[36] A. Kim, Y. Song, M. Kim, K. Lee, and J. H. Cheon. Logistic regression model training based on the approximate homomorphic encryption. *BMC Med Genomics*, 11(Suppl 4):83, Oct 2018.

[37] T. A. Lasko and S. A. Vinterbo. Spectral anonymization of data. *IEEE Trans Knowl Data Eng*, 22(3):437–446, Mar 2010.

[38] E. Lee, J.-W. Lee, J. Lee, Y.-S. Kim, Y. Kim, J.-S. No, and W. Choi. Low-complexity deep convolutional neural networks on fully homomorphic encryption using multiplexed parallel

convolutions. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 12403–12422. PMLR, 17–23 Jul 2022.

[39] Y. Lindell. Secure multiparty computation. *Commun. ACM*, 64(1):86–96, dec 2020.

[40] V. Lyubashevsky, C. Peikert, and O. Regev. On ideal lattices and learning with errors over rings. In *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2010.

[41] C. Marcolla, V. Sucasas, M. Manzano, R. Bassoli, F. H. P. Fitzek, and N. Aaraj. Survey on fully homomorphic encryption, theory, and applications. *Proceedings of the IEEE*, 110(10):1572–1609, 2022.

[42] D. Miao, C. A. Margolis, W. Gao, M. H. Voss, W. Li, D. J. Martini, C. Norton, D. é, S. M. Wankowicz, D. Cullen, C. Horak, M. Wind-Rotolo, A. Tracy, M. Giannakis, F. S. Hodi, C. G. Drake, M. W. Ball, M. E. Allaf, A. Snyder, M. D. Hellmann, T. Ho, R. J. Motzer, S. Signoretti, W. G. Kaelin, T. K. Choueiri, and E. M. Van Allen. Genomic correlates of response to immune checkpoint therapies in clear cell renal cell carcinoma. *Science*, 359(6377):801–806, Feb 2018.

[43] C. Mouchet, J. R. Troncoso-Pastoriza, J. Bossuat, and J. Hubaux. Multiparty homomorphic encryption from ring-learning-with-errors. *Proc. Priv. Enhancing Technol.*, 2021(4):291–311, 2021.

[44] P. Naidoo, C. Bouharati, V. Rambiritch, N. Jose, S. Karamchand, R. Chilton, and R. Leisegang. Real-world evidence and product development: Opportunities, challenges and risk mitigation. *Wien Klin Wochenschr*, 133(15-16):840–846, Aug 2021.

[45] M. S. Olson. Can real-world evidence save pharma US$1 billion per year? A framework for an integrated evidence generation strategy. *J Comp Eff Res*, 9(2):79–82, Jan 2020.

[46] PALISADE Lattice Cryptography Library (release 1.11.9). `https://palisade-crypto.org/`, Dec 2022.

[47] J. P. Pirruccello, M. D. Chaffin, E. L. Chou, S. J. Fleming, H. Lin, M. Nekoui, S. Khurshid, S. F. Friedman, A. G. Bick, A. Arduini, L. C. Weng, S. H. Choi, A. D. Akkad, P. Batra, N. R. Tucker, A. W. Hall, C. Roselli, E. J. Benjamin, S. K. Vellarikkal, R. M. Gupta, C. M. Stegmann, D. Juric, J. R. Stone, R. S. Vasan, J. E. Ho, U. Hoffmann, S. A. Lubitz, A. A. Philippakis, M. E. Lindsay, and P. T. Ellinor. Deep learning enables genetic analysis of the human thoracic aorta. *Nat Genet*, 54(1):40–51, Jan 2022.

[48] S. Sav, A. Pyrgelis, J. R. Troncoso-Pastoriza, D. Froelicher, J. Bossuat, J. S. Sousa, and J. Hubaux. POSEIDON: privacy-preserving federated neural network learning. In *28th Annual Network and Distributed System Security Symposium, NDSS 2021, virtually, February 21-25, 2021.* The Internet Society, 2021.

[49] Barriers and disincentives to the use of real-world evidence and real-world data. In C. Shore, A. W. Gee, B. Kahn, and E. H. Forstag, editors, *National Academies of Sciences, Engineering, and Medicine; Health and Medicine Division; Board on Health Sciences Policy; Forum on Drug Discovery, Development, and Translation.* National Academies Press, Washington, DC, 2019.

[50] Y. Wu, E. M. Byrne, Z. Zheng, K. E. Kemper, L. Yengo, A. J. Mallett, J. Yang, P. M. Visscher, and N. R. Wray. Genome-wide association study of medication-use and associated disease in the UK Biobank. *Nat Commun*, 10(1):1891, Apr 2019.

[51] A. C.-C. Yao. How to generate and exchange secrets. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 162–167, 1986.

Appendices

A Summary

The appendices describe a threshold FHE variant of the CKKS scheme; provide details for interactive 2-party bootstrapping, more general interactive bootstrapping, and interactive private join-and-compute protocols along with the security proofs; provide numerical survival analysis results for the colorectal cancer data set; describe the multiparty setup used for the public data set experiments; provides detailed accuracy results of FHE computations for descriptive statistics, survival analysis, and logistic regression training using the public data set [9]; and illustrate the effect of polynomial degree and range in approximating the sigmoid function. A technical overview of our contributions that complements the higher-level overview and contributions presented in the main text is given below.

We provide a private, accurate and efficient solution for collaborative analysis of healthcare data using a synergy of the main cryptographic tools for privacy-preserving computation: fully homomorphic encryption and secure multiparty computation. Fully homomorphic encryption enables the non-interactive evaluation of polynomials over encrypted data. While the evaluation of fixed-degree polynomials (arithmetic circuits of fixed depth) is efficient, the bootstrapping operation which enables further encrypted computations after exhausting the initial fixed depth is memory- and compute-intensive. Alternatively, multiparty computation enables the interactive evaluation of polynomials over secret-shared data. The local computations are efficient, but parties need to use an all-to-all communication pattern to jointly evaluate multiplications and decryption. We use a threshold version of fully homomorphic encryption, such that multiple data owners can non-interactively encrypt and compute over their data, limiting the interaction only to distributed key generation, bootstrapping, and decryption. In this context, we build on the general multiparty interactive bootstrapping protocol proposed in [48], where parties add a Gaussian noise sample to each ciphertext component and a statistical additive mask. We observe that privacy for the intermediate partial decryptions can be ensured using a single smaller noise sample, and suggest an optimization for the general bootstrapping procedure based on this. For the two-party case, we propose a novel optimized two-party bootstrapping protocol, where the parties jointly perform a distributed rounding step that ensures the obtained shares add up to the correct message over the integers rather than with modular reduction; our two-party method does not require a statistical mask. This novel protocol achieves smaller parameters compared to the general multiparty bootstrapping protocol instantiated for two parties, which reduces the cost of the whole encrypted computation (of which the interactive bootstrapping is a step). We formally prove the security of this protocol via the ideal/real-world paradigm.

To address scenarios where data owners can hold records for common individuals, we propose a privacy-preserving solution for the private join collaboration model. Concretely, we aim to obtain the encrypted data records corresponding to the identifiers in the intersection of all data sets at a computing party, which then homomorphically evaluates the desired computation. The main idea of the solution is that data owners permute and hash their (padded) identifiers and transmit these hashes to the other data owners, each of which repeatedly applies a commutative cipher. After as many communication rounds as the number of the data owners, the computing party can identify which hashes belong to all sets, but cannot retrieve which identifiers these hashes correspond to. For the last step where the computing party should obtain the encrypted records corresponding to the intersection, we describe different versions that trade the privacy of the intersection identifiers with respect to the data owners (but they are never revealed to the computing party) with the overall communication and computation efficiency. The first version, which returns the hashed identifiers in the intersection to the data owners, leads to a more efficient compute part, while the second version hides the intersection identifiers from the data owners but requires more homomorphic computations by the computing party in the subsequent compute part. Intermediate versions are also possible. Our protocol extends the solution in [33], where the computation happens between only one data owner (which has the data records corresponding to the identifiers) and a computing party (that has some identifiers).

Finally, due to the restrictions on the computation types allowed by fully homomorphic encryption (arithmetic circuits or polynomials) and for efficiency purposes, we have to approximate the desired functionality. In particular, there are two major changes that need to be performed compared to the cleartext computations: (i) approximate the nonlinear functions via polynomials, which we do via Chebyshev interpolation and which requires fixing a degree of the approximation polynomial and an interval on which to evaluate the approximation, and (ii) fix an optimization algorithm and a number of iterations (testing convergence would leak information about the private data). Apart from the approximation errors introduced by these two changes, the CKKS fully homomorphic encryption scheme is approximate, meaning that deep computations can introduce errors in the least significant bits. We discuss and justify our choices for the algorithms and parameters mentioned above and describe in depth the differences between the results obtained after decryption and the cleartext results. We emphasize that we preferred generality—obtaining accuracy with the same parameters on multiple data sets and not overfitting the choices on each data set—at the expense of a larger approximation degree and interval, and more iterations in optimization algorithms.

B Background

We denote scalars as lowercase letters, vectors as lowercase boldface letters, and matrices as uppercase boldface letters. A function $f : \mathbb{N} \to [0, 1]$ is negligible if for every polynomial p, there exists a positive scalar m such that $f(n) < 1/p(n)$ for all $n \geq m$. We abuse notation and write $f(\kappa) < \mathsf{negl}(\kappa)$ when it is in the set of negligible functions, where κ denotes the computational security parameter. Similarly, we let $\mathsf{poly}(\kappa)$ denote a function which is bounded by some polynomial in κ. For a finite set S, sampling it uniformly at random is denoted as $s \leftarrow S$. We denote general sampling as $x \leftarrow X_\kappa$ where X_κ is a distribution parameterized by κ. We say two distribution ensembles, $\{X_\kappa\}, \{Y_\kappa\}$, are computationally indistinguishable, denoted $X_\kappa \approx_c Y_\kappa$, if for all sufficiently large security parameters, $\kappa > 0$, for all non-uniform probabilistic polynomial time distinguishers, D, it holds that $|Pr_{x \leftarrow X_\kappa}(D(1^\kappa, x) = 1) - Pr_{y \leftarrow Y_\kappa}(D(1^\kappa, y) = 1)| < \mathsf{negl}(\kappa)$.

We will use the Decisional-Diffie-Hellman (DDH) assumption, defined below. The assumption depends on the representation of the group. For example, we will use elliptic curve groups.

Definition 1 (DDH)**.** *Let G be a finite (multiplicative) cyclic group of order p with generator g. The DDH assumption states that: $\{(g, g^a, g^b, g^{ab})|\ g \leftarrow G, a, b \leftarrow \mathbb{Z}_p\} \approx_c \{(g, g^a, g^b, g^c)|\ g \leftarrow G, a, b, c \leftarrow \mathbb{Z}_p\}$.*

We now recall the multiparty security definition. We consider a semi-honest adversarial model, which means that the parties controlled by the adversary follow the protocol steps honestly, but can process and log the transcript (all the messages received and used in the execution) of the protocol in order to try to extract more information than specified by the protocol.

Informally, a protocol is said to be secure in the semi-honest model if no information is leaked to a party participating in the protocol apart from what can be obtained solely from that party's prescribed input and prescribed output. In other words, all information arising from the intermediate computations is being concealed from any participating party. The same intuition can be extend to coalitions of colluding parties.

Definition 2 (Protocol security [28])**.** *Let $f : (\{0,1\}^*)^k \to (\{0,1\}^*)^k$ be a k-ary (potentially randomized) functionality where $f_i(\mathbf{x})$ is party P_i's output, for $i \in \{1,\dots,k\}$, x_i is party P_i's input, and $\mathbf{x} = (x_1,\dots,x_k)$ denotes all the parties' inputs. For $I = \{i_1,\dots,i_t\} \subset \{1,\dots,k\}$, which models a coalition of parties, we let $f_I(\mathbf{x})$ denote the subsequence $(f_{i_1}(\mathbf{x}),\dots,f_{i_t}(\mathbf{x}))$. Let Π be an k-party protocol computing the ideal functionality f. Let View_i denote party P_i's view in the real execution of the protocol Π, consisting of its inputs, messages received in the protocol and randomness sampled. The view of the coalition is denoted as $\mathsf{View}_I := (I, \mathsf{View}_{i_1},\dots,\mathsf{View}_{i_t})$. Let $\mathsf{output}(\mathbf{x})$ denote the output of the real execution of Π on the parties' inputs $\mathbf{x}$.*

The protocol Π computing f is said to be secure in the semi-honest model if for all sufficiently large security parameters $\kappa > 0$, and for any allowed set of parties $I \subset \{1,\dots,k\}$, there exists a simulator Sim_I such that:

$$\{(\mathsf{Sim}_I(x_I, f_I(\mathbf{x})), f(\mathbf{x}))\}_{\mathbf{x}} \approx_c \{(\mathsf{View}_I(\mathbf{x}), \mathsf{output}(\mathbf{x}))\}_{\mathbf{x}},$$

for all $x_1,\dots,x_k \in \{0,1\}^{\mathsf{poly}(\kappa)}$.

A common specialization of this definition is the two-party case, where a simulator is built for each party and there are no collusions.

C Threshold CKKS Encryption

The first scheme we describe is the single-key CKKS encryption scheme and then its threshold version [3,14].

Cheon-Kim-Kim-Song Homomorphic Encryption Our solution is based on the CKKS approximate FHE scheme [14]. CKKS is an RLWE-based scheme which supports floating point-like Single Instruction Multiple Data (SIMD) multiplication. Let N be a power two (often between 8192 and 131072). Then, the polynomial ring $R := \mathbb{Z}[X]/(X^N + 1)$ is the $2N$-th cyclotomic field's ring of integers. Let $R_Q := R/QR$ be the ring with coefficients reduced modulo Q. We use a modulus which supports the number theoretic transform (NTT), where $R_{Q_i} := R/Q_iR$ is the ring with coefficients reduced modulo $Q_i = q_0q_1\cdots q_i$, $Q_L = q_0q_1\cdots q_L$ is the largest modulus, with L multiplicative levels available, and $q_i = 1 \pmod{2N}$ for all $i = 0,\dots,L$. Secret keys are $s \in R$ with balanced, ternary coefficients, $s \in \{0,\pm1\}^N \subset R$. We denote the uniform distribution over ternary polynomials as χ_s. Further, χ_{DG} denotes the discrete Gaussian over R with coefficients sampled independently with standard deviation 3.19 [2]. Distributed decryption uses a noise-flooding distribution χ_{fld}, which is 20 bits larger than the expected ciphertext error. The CKKS scheme encodes messages with a scaling factor, together with a packing over the real numbers: $\mathsf{m} = \mathsf{encode}(\mathbf{m})$ where encode first scales up the message by a scaling factor Δ, computes the inverse discrete Fourier transform (DFT), then rounds to the integers modulo Q_L. This packing allows us to perform homomorphic SIMD addition, SIMD multiplication with rescaling (floating point-like multiplication), and rotation of the vector of scalars packed in the plaintexts/ciphertexts. Our solution is based on an optimized variant by Kim et al. [34] of the Residue Number System (RNS) representation, also called the double Chinese Remainder Theorem (CRT) from Gentry et al. [26].

The main algorithms of the CKKS scheme are:

- $\mathsf{Setup}(1^\kappa, L)$. For an input level L, set the ring dimension N, scaling factor Δ, RNS modulus $Q_L = q_0 \cdots q_L$ with $q_i = 1 \pmod{2N}$ prime, and the distributions χ_s and χ_{DG}. Return these parameters $\mathsf{pp} = (L, \Delta, N, Q_L, \{q_i\}, \chi_s, \chi_{DG})$. (We assume all remaining algorithms have access to pp.)

- $\mathsf{Keygen}(\mathsf{pp})$. Sample a secret key $s \leftarrow \chi_s$, a uniformly random $a \in R_{Q_L}$, and an error $e \leftarrow \chi_{DG}$. Return $\mathsf{sk} := s$ and $\mathsf{pk} := (a, b) = (a, -as + e) \in R_{Q_L}^2$.

- $\mathsf{KSGen}(\mathsf{sk}, \mathsf{sk}')$. This algorithm generates key-switching keys from $\mathsf{sk}' = s'$ to $\mathsf{sk} = s$. These keys are given as polynomials in a larger modulus PQ_L where P is an NTT friendly integer the same bit-length of Q_L. Sample $a' \leftarrow R_{PQ_L}$ uniformly at random, $e \leftarrow \chi_{DG}$, return $\mathsf{swk} := (a', b') = (a', -a's' + e + Ps') \in R_{PQ_L}^2$. This algorithm is used to compute the relinearization key $\mathsf{evk} \leftarrow \mathsf{KSGen}(s, s^2)$ and the rotation keys $\mathsf{rk}^{(\sigma)} \leftarrow \mathsf{KSGen}(s, \sigma(s))$ where σ is a rotation (automorphism).

- $\mathsf{Enc}_{\mathsf{pk}}(\mathsf{m})$. For a message polynomial $\mathsf{m} = M(X) \in R$, sample $v \leftarrow \chi_s$, $e_0, e_1 \leftarrow \chi_{DG}$, and return $\mathsf{ct} := v \cdot \mathsf{pk} + (e_0, e_1 + \mathsf{m}) \in R_{Q_L}^2$.

- $\mathsf{encode}(\Delta, \mathbf{v})$. Given a real vector $\mathbf{v} \in \mathbb{R}^{N/2}$, compute $\mathsf{m} = M(X) := \left\lceil \tau^{-1}(\Delta \cdot \mathbf{v}) \right\rfloor \in R$ where τ is the canonical embedding on R (see Lyubashevsky et al. [40]).

- $\mathsf{decode}(\Delta, \mathsf{m})$. Given a message $\mathsf{m} = M(X) \in R$, compute $\mathbf{v} := \tau(M(X))/\Delta \in \mathbb{R}^{N/2}$.

- $\mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct})$. Given $\mathsf{ct} = (a, b) \in R_{Q_i}^2$, return $\hat{\mathsf{m}} := as + b \pmod{Q_i}$.

- $\mathsf{CAdd}(\mathsf{ct}, \alpha)$. Given a constant $\alpha \in R$ and a ciphertext ct, return $\mathsf{ct} + (0, \alpha)$.

- $\mathsf{CMult}(\mathsf{ct}, \alpha)$. Given a constant $\alpha \in R$ and a ciphertext ct, return $\alpha \cdot \mathsf{ct}$.

- $\mathsf{Add}(\mathsf{ct}_1, \mathsf{ct}_2)$. Given two ciphertexts at the same level, $\mathsf{ct}_1, \mathsf{ct}_2 \in R_{Q_i}^2$, return $\mathsf{ct}_1 + \mathsf{ct}_2$.

- $\mathsf{Mult}_{\mathsf{evk}}(\mathsf{ct}_1, \mathsf{ct}_2)$. For $\mathsf{ct}_i = (a_i, b_i) \in R_{Q_j}^2$, let $(d_0, d_1, d_2) = (a_1a_2, a_1b_2 + a_2b_1, b_1b_2) \in R_{Q_j}^3$. Return $\mathsf{ct}_\times = (d_0, d_1) + \left\lceil \tilde{Q}^{-1} d_2 \cdot \mathsf{evk} \right\rfloor$ where $\tilde{Q} = PQ'$ is such that $Q_j = Q_L/Q'$.

- $\mathsf{Rotate}_{\mathsf{rk}^{(\sigma)}}(\mathsf{ct}, \sigma)$. For $\mathsf{ct} = (a, b) \in R_{Q_j}^2$ and rotation σ, output $\mathsf{ct}_{\text{rotate}} \leftarrow (0, \sigma(b)) + \left\lceil \tilde{Q}^{-1} \cdot \sigma(a) \cdot \mathsf{rk}^{(\sigma)} \right\rfloor \pmod{Q_j}$ where $\tilde{Q} = PQ'$ is such that $Q_j = Q_L/Q'$.

- $\mathsf{Rescale}(\mathsf{ct})$. For a ciphertext $\mathsf{ct} \in R_{Q_L}^2$, $Q_j = q_j q_{j-1} \cdots q_0$, and an integer k, output $\mathsf{ct}' \leftarrow \left\lceil q_j^{-1} \cdot \mathsf{ct} \right\rfloor \pmod{Q_{j-1}}$ for $Q_{j-1} = Q_j/q_j$.

Secret-Shared Threshold CKKS We use distributed key generation in the threshold FHE setting where $\mathsf{sk} := s = s_0 + \cdots + s_{k-1}$ and party P_i holds secret share $s_i \leftarrow \chi_s$ [3, 43].

- $\mathsf{DKeygen}(\mathsf{pp}, a, \{s_i\})$. Each party P_i has a secret share $s_i \leftarrow \chi_s$ together with a common random polynomial $a \in R_{Q_L}$. They each sample $e_i \leftarrow \chi_{DG}$. Send $b_i = -as_i + e_i \in R_{Q_L}$ to some aggregating party. This party returns $\mathsf{pk} = (a, \sum_i b_i)$.

- $\mathsf{DKSGen}(\mathsf{pp}, a, \sigma, \{s_i\})$. Given a common random polynomial $a \in R_{PQ_L}$ and a rotation σ, each party P_i samples $e_i \leftarrow \chi_s$ and sends $b_i = -as_i + e_i + \sigma(s)P$. An aggregating party returns $\mathsf{rk} := (a, \sum_i b_i) \in R^2_{PQ_L}$ as the key-switching key for σ.

- $\mathsf{DRelinGen}(\mathsf{pp}, a, \{s_i\})$. On input RLWE parameters, a common random polynomial $a \in R_{Q_L}$, and a secret share s_i, party P_i does the following in two rounds:

  1. Samples $e_i \leftarrow \chi_{DG}$ and sends $h_{1,i} := -as_i + Ps_i + e_i \in R_{PQ_L}$ to an aggregating party. This party computes $(a, b) = (a, h_1) := (a, \sum h_{1,i})$ then sends it back to P_i. Notice that (a, b) is an encryption of $P \sum s_i$ under $s = \sum s_i$.
  2. Given (a, b), P_i samples $e_i^0, e_i^1 \leftarrow \chi_{DG}$, then returns $(a_i, b_i) := (s_i a + e_i^0, s_i b + e_i^1)$ to the aggregating party. The aggregating party returns $\mathsf{evk} := \sum_i (a_i, b_i)$. The output evk is an encryption of $\sum_i s_i s P = s^2 P$ for $s = \sum_i s_i$ under the joint key $s = \sum_i s_i$.

- $\mathsf{DDec}(\mathsf{ct}, \{s_i\})$: Given a $\mathsf{ct} \in R^2_{Q_0}$ (rescale if needed), each party besides some aggregating party, say P_0, samples a flooding error $e_i \leftarrow \chi_{fld}$ and generates its partial decryption share $p_i := as_i + e_i$. It sends its share to the aggregator P_0. Then, the aggregating party returns $b + as_0 + \sum p_i \pmod{Q_0}$.

The distributed key generation, key-switching key generation, relinearization key generation, and the distributed decryption protocols were initially developed and proved for the LWE setting by Asharov et al. [3]. The RLWE versions were given by [43, 48] with security from RLWE together with statistical noise flooding in distributed decryption.

We will also use an exact (leveled) fully homomorphic encryption scheme for integer computations such as frequency computation. In particular, we use the BFV scheme [21], which is also lattice-based, supports SIMD packing and its security is based on the RLWE hardness assumption. Given the similarity to the CKKS scheme, we only mention the encryption and decryption algorithms here. Here, t denotes the plaintext modulus.

- $\mathsf{BFV.Enc}_{\mathsf{pk}}(\mathsf{m})$. For a message polynomial $\mathsf{m} = M(X) \in R$, sample $v \leftarrow \chi_s$, $e_0, e_1 \leftarrow \chi_{DG}$, and return $\mathsf{ct} := v \cdot \mathsf{pk} + (e_0, e_1 + \lfloor q/t \rceil \mathsf{m}) \in R^2_{Q_L}$.

- $\mathsf{BFV.Dec}_{\mathsf{sk}}(\mathsf{ct})$. Given $\mathsf{ct} = (a, b) \in R^2_{Q_i}$, return $\hat{\mathsf{m}} := \lfloor (t/Q_i) \cdot (as + b \pmod{Q})_i) \rceil \pmod{t}$.

D Two-Party Interactive Bootstrapping

In order to continue computations over exhausted ciphertexts (ciphertexts that have accumulated too much noise from previous computations), a *bootstrapping* procedure needs to be performed, which refreshes the ciphertexts. However, classical FHE (single-party) bootstrapping implies a substantial computational and memory overhead, as described in the main paper.

To avoid this overhead, we propose a novel protocol for two-party interactive bootstrapping for the threshold CKKS scheme. We remark that this protocol can be slightly modified to support other RLWE-based threshold FHE schemes such as threshold BFV. The goal is for two parties, a Client and a Server, to jointly bootstrap an exhausted ciphertext that has a large noise component and generate an equivalent ciphertext that is encrypting the same plaintext message, with a much smaller noise. More exactly, the Server is given a CKKS ciphertext $(a', b') \in R_q^2$ encrypting an encoded message $c = \mathsf{Dec}_{\mathsf{sk}}(a', b') = a'(s_0 + s_1) + b'$ such that $\|c\|_\infty < \beta$, and it should compute a fresh CKKS encryption $\mathsf{Enc}_{\mathsf{pk}}(c)$ of the same (encoded) message c under public key pk with the help of the Client. The ideal functionality of the two-party interactive bootstrapping is given in Functionality 1.

Functionality 1 Ideal Functionality for 2-party (Client and Server) interactive bootstrapping protocol Π: $\mathcal{F}_{\mathcal{IB}}$

Input: A ciphertext $\mathsf{ct}' = (a', b') \in R_q^2$ encrypted under a pk (with secret key $\mathsf{sk} = s_0 + s_1$), and each party's secret share $s_i \in R_q$.

Output: A fresh encryption $\hat{\mathsf{ct}}$ under a public key pk, output to the Server.

```
1: Assemble the joint secret key from the inputs $\mathsf{sk} = s_0 + s_1$.
2: Decrypt $\mathsf{m} \leftarrow \mathsf{Dec}_{\mathsf{sk}}(\mathsf{ct}')$.
3: Sample encryption randomness $r$ and encrypt $\hat{\mathsf{ct}} \leftarrow \mathsf{Enc}_{\mathsf{pk}}(\mathsf{m}; r)$.
4: return $\hat{\mathsf{ct}}$ to the Server and nothing to the Client.
```

Our proposed protocol is given in Protocol 2. The main idea underlying the protocol is distributed rounding, where each party uses its secret key share to compute a partial decryption modulo q, then with high probability, they coordinate in adding (subtracting) $q/2$. This ensures that the rounded partial decryptions add up to the correct message over the integers without taking the sum modulo q. The intuition behind the interactive rounding is that, if the encrypted value is much smaller than q, the unmasked decryption shares are close to the negations of each other. Therefore, they are close to being symmetric around 0 in $\{-(q-1)/2, \ldots, 0, \ldots, (q-1)/2\}$. In the case that they are closer to 0 than $q/2$, addition can be done without reducing modulo q. If not, then one modular reduction is needed and we can compute this locally. The parties avoid further modular reduction in both cases.

We use a hash function $\mathsf{H} : R_Q^2 \times R_q \to R_q$ modeled as a random oracle (RO). We remark that random oracles are not needed for security, but for correctness. What we require is that $\mathsf{H}(\mathsf{pk}, a)$ is distributed uniformly at random over R_q for statistical purposes.

Further, note that $\mathsf{Enc}_{\mathsf{pk}}$ may be a CKKS encryption using a larger modulus Q or any other linearly homomorphic public key encryption scheme, as the computation performed by the parties uses $\mathsf{Enc}_{\mathsf{pk}}$ as a black box. In particular, the method can be used not only to increase the plaintext modulus of a ciphertext, but also to switch between different encryption keys, or even between different schemes. The latter operation may require some additional operations to take into account different encodings used by CKKS and other schemes. We use the same CKKS public key for the input and output ciphertext in the following.

Protocol 2 2-Party Interactive Bootstrapping Protocol $\Pi_{\mathcal{IB}}$

Input: Each party has a secret key share $s_i \in R_q$. The Server additionally has a ciphertext $\mathsf{ct}' = (a', b') \in R_q^2$ encrypted under a pk (with secret key $\mathsf{sk} = s_0 + s_1$) encrypting m.

Output: A fresh encryption $\hat{\mathsf{ct}} \in R_Q^2$ under a pk, potentially unrelated to sk, encrypting m.

```
1: The Server samples and adds a public-key encryption of 0: $(a, b) = (a', b') + \mathsf{Enc}_{\mathsf{pk}}(0) \in R_q^2$. Server sends $a$ to the Client.
2: Both parties compute $\rho \leftarrow \mathsf{H}(\mathsf{pk}, a)$.
3: Server computes $c_0 = as_0 + b + \rho \pmod q$ and the rounding: If $|c_0| > q/4$, then $c_0 \leftarrow c_0 + (q/2) \pmod q$. The operation is performed coordinate-wise, on each coefficient of $c_0$ independently.
4: Client computes $c_1 = as_1 - \rho \bmod q$ and the rounding: If $|c_1| > q/4$, then $c_1 \leftarrow c_1 + (q/2) \pmod q$. The operation is performed coordinate-wise, on each coefficient of $c_1$ independently.
5: Client then encrypts $c_1$ as $\mathsf{ct}_C \leftarrow \mathsf{Enc}_{\mathsf{pk}}(c_1) \in R_Q^2$, and sends it to the Server.
6: Server computes $\mathsf{ct}_S \leftarrow \mathsf{Enc}_{\mathsf{pk}}(c_0) \in R_Q^2$ and returns $\mathsf{ct} = \mathsf{ct}_C + \mathsf{ct}_S \in R_Q^2$.
```

In order to achieve overwhelming correctness according to Theorem 1, interactive bootstrapping should be performed before the encrypted message gets too large relative to the current level's modulus q. This can be easily achieved by increasing the ciphertext modulus q by an RNS limb or two. Note that the only non-arithmetic operation performed by the protocol is the check $|c_i| > q/4$ in step 5, and it is performed in the clear. Therefore, we can implement a correct interactive bootstrapping protocol very efficiently in CRT representation, by simply adding a limb to the modulus q.

We compare the ciphertext modulus q required in this novel two-party interactive bootstrapping protocol with the modulus required in a general multiparty interactive bootstrapping protocol, presented subsequently. We note that this comparison holds for the RNS representation using 64-bit native words, which offers the best performance in practice (in PALISADE and other common FHE libraries). In the latter protocol, for two parties and a statistical security parameter of 128 bits, we obtain that interactive bootstrapping requires the ciphertext modulus q to increase by three extra CRT limbs, i.e., by 180-bits. On the other hand, for the current two-party interactive bootstrapping protocol, for a correctness probability of 2^{-50}, and for the range of ring modulus dimensions of interest we obtain that the ciphertext modulus q increases only by one CRT limb, i.e., by 60 bits.

We now formally state and prove the correctness and the security of Protocol 2 with respect to Definition 2.

Theorem 1. *Let $\mathsf{ct}' = (a', b') \in R_q^2$ be a CKKS ciphertext with $\|a'(s_0 + s_1) + b' = c\|_\infty \le \beta \ll q$. The above protocol is correct except with probability at most $2N\beta/q$ over the randomness of H, modeled as a random oracle.*

Proof. We claim that if each coefficient of c_i satisfies $|c_i| \notin (q/4 \pm \beta)$, then the protocol is correct. To see this, compute

$$c_0 + c_1 = (as_0 + b + \rho) + as_1 - \rho = a(s_0 + s_1) + b = c \pmod q.$$

We know that $\|c\|_\infty < \beta$ by assumption, so, the two values $c_0, -c_1$ are within distance β from each other. Assuming they are not in $[-q/4-\beta, -q/4+\beta]$ or in $[q/4-\beta, q/4+\beta]$, they will both satisfy the condition $|c_i| > q/4$ or they will both violate the condition $|c_i| > q/4$. It follows that in steps 3 and 4, the value $(q/2)$ is added to both of them, or to neither of them. In either of these cases, the sum $c_0 + c_1 \pmod q$ does not change, and in particular $c_0 + c_1 = c \pmod q$.

Moreover, we also know that (before step 3) $|c_i| < q/2$ because c_i is reduced modulo q. Therefore, after step 4, it will satisfy $|c_i| < q/4$. So, the sum $|c_0 + c_1| < q/2$ is already reduced modulo q after step 4, and we have $c_0 + c_1 = c$ over the integers. Since $\mathsf{Enc}_{\mathsf{pk}}$ is linearly homomorphic, the final output satisfies

$$\mathsf{Enc}_{\mathsf{pk}}(c_0) + \mathsf{Enc}_{\mathsf{pk}}(c_1) = \mathsf{Enc}_{\mathsf{pk}}(c_0 + c_1) = \mathsf{Enc}_{\mathsf{pk}}(c)$$

and the protocol is correct.

It remains to bound the failure probability. Each coefficient of c_i is distributed uniformly at random over $\mathbb{Z}_q$ because $\rho \leftarrow \mathsf{H}(\mathsf{pk}, a)$ was added to it. Therefore,

$$\Pr[\ c_i \in [-q/4-\beta, -q/4+\beta] \cup [q/4-\beta, q/4+\beta]\ ] \leq 4\beta/q.$$

Taking a union bound over all N coefficients, we get that the probability of any of them being too close to $\pm q/4$ is at most $4\beta N/q$.

We can be reduce the failure probability to $2\beta N/q$ in a straightforward manner, by considering the sign of c, and analyzing the cases $c > 0$ and $c < 0$ separately. □

Theorem 2. *The protocol $\Pi_{\mathcal{IB}}$ is secure in the semi-honest model assuming that RLWE ciphertexts are pseudorandom under secret-shared secret keys, given all but one share, achieving the functionality described in Functionality 1.*

Proof. The adversary is given oracle access to the random oracle by mimicking an RO with a randomly computed function from $R_Q^2 \times R_q$ to R_q. Every time the adversary queries $\mathsf{H}(x, y)$ on $(x, y) \in R_Q^2 \times R_q$, we look up the table to see if (x, y) has a value in the table. If so, return this value. Otherwise, compute a random $\rho \in R_q$, store it in an entry labeled by (x, y) in the table, and return ρ to the adversary.

We first specify the inputs, outputs and views of the Client and the Server.
$\mathsf{Input}_C^\Pi : s_1$.
$\mathsf{Output}_C^\Pi : \emptyset$.
$\mathsf{View}_C^\Pi : (s_1, \mathsf{coins}_C; a)$.
In the Client's view, a is received from the Server and the random coins_C are the sampled randomness used by the Client (to encrypt $\mathsf{Enc}_{\mathsf{pk}}(c_1)$ in line 6).

$\mathsf{Input}_S^\Pi : s_0, (a', b')$.
$\mathsf{Output}_S^\Pi : \mathsf{ct}$.
$\mathsf{View}_S^\Pi : (s_0, (a', b'), \mathsf{coins}_S; \mathsf{Enc}_{\mathsf{pk}}(c_1))$.
In the Server's view, b and $\mathsf{Enc}_{\mathsf{pk}}(c_1)$ are received from the Client, while coins_S are the sampled randomness used by the Server (to create (a, b) in line 1 and to encrypt $\mathsf{Enc}_{\mathsf{pk}}(c_0)$ in line 7).

Both the Client and Server also receive as public input the public key pk (RLWE encryption of 0) corresponding to the joint secret key $\mathsf{sk} = s_0 + s_1$. Note that the Client does not receive the output ciphertext.

The simulator for the Server is as follows:
$\underline{\mathsf{Sim}_S((s_0, \mathsf{ct}'), \hat{\mathsf{ct}})}$:

1. Sample $\mathsf{coins}_S = (\hat{r}_S^0, \hat{r}_S^1)$.
2. $(\hat{a}, \hat{b}) \leftarrow \mathsf{ct}' + \mathsf{Enc}_{\mathsf{pk}}(0; \hat{r}_S^0)$.
3. $\hat{\rho} \leftarrow \mathsf{H}(\mathsf{pk}, \hat{a})$.
4. Compute $\hat{c}_0 \leftarrow \left\lceil \hat{b} + \hat{a}s_0 + \hat{\rho} \right\rfloor_{dist}$.
5. $\hat{\mathsf{ct}}_C \leftarrow \hat{\mathsf{ct}} - \mathsf{Enc}_{\mathsf{pk}}(\hat{c}_0; \hat{r}_S^1)$.
6. Return $\left(s_0, \mathsf{ct}', (\hat{r}_S^0, \hat{r}_S^1), \hat{\mathsf{ct}}_C\right)$.

The joint transcript from the simulated view and the ideal functionality is

$$((s_0, \mathsf{ct}', (\hat{r}_S^0, \hat{r}_S^1), \hat{\mathsf{ct}} - \mathsf{Enc}_{\mathsf{pk}}(\hat{c}_0; \hat{r}_S^1)), \hat{\mathsf{ct}}), \text{ for } \hat{\mathsf{ct}} \leftarrow \mathcal{F}_{\mathcal{IB}}(s_0, s_1, \mathsf{ct}').$$

We construct a sequence of hybrids to prove indistinguishability between the ideal-world distribution and the real-world distribution.

Assuming the hardness of RLWE for joint keys in the theorem's statement (see Lemma 4 in [3] proved in [4]), the ciphertext $\hat{\mathsf{ct}}$ is indistinguishable from random and is also indistiguishable from $\mathsf{Enc}_{\mathsf{pk}}(\lceil \hat{a}s_1 - \rho \rfloor_{dist}; r_C)$ for some uncorrelated encryption randomness r_C. To see why an encryption of a message depending on a secret key share s_1 still looks pseudorandom to a distinguisher who knows the other share s_0, perform the following computations. Recall that for a ciphertext (d_0, d_1) under the joint secret key $(s_0 + s_1)$, the encrypted message is obtained as $d_1 + d_0(s_0 + s_1)$. Note that $(\hat{a}, 0)$ is an encryption of $\hat{a}(s_0 + s_1)$ and $(0, \hat{a}s_0)$ is an encryption of $\hat{a}s_0$. Therefore, starting from an encryption of zero, we can write

$$\mathsf{Enc}_{\mathsf{pk}}(\hat{a}s_1; r_C) = \mathsf{Enc}_{\mathsf{pk}}(0; r_C) + (\hat{a}, 0) + (0, \hat{a}s_0).$$

The above shows that $\mathsf{Enc}_{\mathsf{pk}}(\hat{a}s_1; r_C)$ is sampleable given s_0. Since $\mathsf{Enc}_{\mathsf{pk}}(0; r_C)$ is indistinguishable from random, then $\mathsf{Enc}_{\mathsf{pk}}(\hat{a}s_1; r_C)$ is also indistinguishable from random, for an unknown r_C. Furthermore, this is also indistinguishable from $\mathsf{Enc}_{\mathsf{pk}}(\lceil \hat{a}s_1 - \hat{\rho} \rfloor_{dist}; r_C)$ and thus, it can replace $\hat{\mathsf{ct}}$ in the view.

The hybrids are:

$$\begin{aligned}
&\{((s_0, \mathsf{ct}', (\hat{r}_S^0, \hat{r}_S^1), \hat{\mathsf{ct}} - \mathsf{Enc}_{\mathsf{pk}}(\hat{c}_0; \hat{r}_S^1)), \hat{\mathsf{ct}})\} \approx_c \\
&\{((s_0, \mathsf{ct}', (\hat{r}_S^0, \hat{r}_S^1), \mathsf{Enc}_{\mathsf{pk}}(\lceil \hat{a}s_1 - \hat{\rho} \rfloor_{dist}; r_C) - \mathsf{Enc}_{\mathsf{pk}}(\hat{c}_0; \hat{r}_S^1)), \mathsf{Enc}_{\mathsf{pk}}(\lceil \hat{a}s_1 - \hat{\rho} \rfloor_{dist}; r_C))\} \approx_c \\
&\{((s_0, \mathsf{ct}', (\hat{r}_S^0, \hat{r}_S^1), \mathsf{Enc}_{\mathsf{pk}}(\lceil \hat{a}s_1 - \hat{\rho} \rfloor_{dist}; r_C)), \mathsf{Enc}_{\mathsf{pk}}(\lceil \hat{a}s_1 - \hat{\rho} \rfloor_{dist}; r_C) + \mathsf{Enc}_{\mathsf{pk}}(\hat{c}_0; \hat{r}_S^1))\} \approx_c \\
&\{((s_0, \mathsf{ct}', (r_S^0, r_S^1), \mathsf{Enc}_{\mathsf{pk}}(\lceil as_1 - \rho \rfloor_{dist}; r_C)), \mathsf{Enc}_{\mathsf{pk}}(\lceil as_1 - \rho \rfloor_{dist}; r_C) + \mathsf{Enc}_{\mathsf{pk}}(c_0; r_S^1))\}.
\end{aligned}$$

Note that the last distribution is the actual view of the Server with the real output of the protocol, for the real randomness used by the Server (r_S^0, r_S^1) and the real values a obtained at the Server, which have the same distribution as $(\hat{r}_S^0, \hat{r}_S^1)$ and $\hat{a}$. The indistinguishability holds for all values of inputs.

The simulator for the Client is as follows:
$\underline{\mathsf{Sim}_C(s_1)}$:

1. Sample $\hat{r}_C$.

2. $\hat{a} \leftarrow R_q$.
3. Return $(s_1, \hat{r}_C, \hat{a})$.

The joint distribution of the simulator and the ideal functionality is

$$((s_1, \hat{r}_C, \hat{a}), \hat{\mathsf{ct}}), \text{ for } \hat{\mathsf{ct}} \leftarrow \mathcal{F}_{\mathcal{IB}}(s_0, s_1, \mathsf{ct}').$$

Note that for an uncorrelated, random r', it holds that:

$$\{((s_1, \hat{r}_C, \hat{a}), \hat{\mathsf{ct}})\} = \{((s_1, \hat{r}_C, \hat{a}), \mathsf{Enc}_{\mathsf{pk}}(\mathsf{m}; r'))\}.$$

Further note that for $\hat{\rho} \leftarrow \mathsf{H}(\mathsf{pk}, \hat{a})$, and another uncorrelated r_S, it also holds that

$$\mathsf{Enc}_{\mathsf{pk}}(\mathsf{m}; r') \approx_c \mathsf{Enc}_{\mathsf{pk}}(\lceil b + \hat{a}s_0 + \hat{\rho} \rfloor_{dist}; r_S) + \mathsf{Enc}_{\mathsf{pk}}(\lceil \hat{a}s_1 - \hat{\rho} \rfloor_{dist}; \hat{r}_C),$$

and $\mathsf{Enc}_{\mathsf{pk}}(\mathsf{m}; r')$ is the real output to the protocol. Therefore it holds that the distributions are computationally indistinguishable:

$$\begin{aligned}\{((s_1, \hat{r}_C, \hat{a}), \hat{\mathsf{ct}})\} &= \{((s_1, \hat{r}_C, \hat{a}), \mathsf{Enc}_{\mathsf{pk}}(\mathsf{m}; r'))\} \\ &\approx_c \{((s_1, \hat{r}_C, \hat{a}), \mathsf{Enc}_{\mathsf{pk}}(\lceil b + \hat{a}s_0 + \hat{\rho} \rfloor_{dist}; r_S) + \mathsf{Enc}_{\mathsf{pk}}(\lceil \hat{a}s_1 + \hat{\rho} \rfloor_{dist}; \hat{r}_C))\} \\ &\approx_c \{((s_1, r_C, a), \mathsf{Enc}_{\mathsf{pk}}(\lceil b + as_0 + \rho \rfloor_{dist}; r_S) + \mathsf{Enc}_{\mathsf{pk}}(\lceil as_1 - \rho \rfloor_{dist}; r_C))\}.\end{aligned}$$

The last distribution is the actual view of the Client with the real output of the protocol and for the real randomness used by the Client r_C and for the real a sampled from the same distributions as $\hat{r}_C$ and a. The indistinguishability holds for all values of inputs. □

Practical considerations. For clarity of presentation, we described the interactive bootstrapping protocol for the case where we represent integers modulo q in the balanced form: $\{-(q-1)/2, \ldots, (q-1)/2\}$. On the other hand, our implementation represents integers modulo q in the positive representation: $\{0, 1, \ldots, q-1\}$. This changes the protocol to add $q/2 \pmod q$ when $c_i \in (q/4,\ 3q/4]$.

We note that in practice, a fresh encryption of zero, $\mathsf{Enc}_{\mathsf{pk}}(0)$ is pseudorandom, hence computationally close to a tuple of uniformly random polynomials. This ensures that the coefficients of $c_0 = as_0 + b$ and $c_1 = as_1$ have a distribution observably close to the uniform distribution over $\mathbb{Z}_q$. Therefore, correctness holds with almost the same success probability even if the parties do not use a uniformly random polynomial $\rho \leftarrow \mathsf{H}(\mathsf{pk}, a)$, assuming the pseudorandomness of RLWE ciphertexts.

The interactive bootstrapping protocol in Protocol 2 is intended to be used in a scenario where the Server performs encrypted computations and obtains an exhausted ciphertext (a', b'), but still needs to perform further computations on that ciphertext. The interaction with the Client helps the Server obtain a fresh ciphertext on which the Server can then continue computations. This is the use case described in the main paper. The security proof given for Theorem 2 is for a stand-alone interactive bootstrapping protocol. In the case where the interactive bootstrapping is a subprotocol of a larger protocol where there are subsequent computations at the Server on the refreshed ciphertext, we can further optimize Protocol 2 by having the Server return $\mathsf{ct}_C + (0, c_0)$ in line 6, and still achieve the security of the full protocol. The reason for this is that before the final decryption, secure flooding is always performed (as described in the threshold CKKS decryption), and moreover, the distinguisher in Definition 2 is not given the intermediate output ciphertext of the interactive bootstrapping.

Finally, we remark that the interactive bootstrapping protocol in Protocol 2 can be immediately modified to a symmetric version with two servers that both receive as input the exhausted ciphertext and output a fresh ciphertext of the same underlying message.

E More General Interactive Boostrapping

We now describe a protocol for multiparty interactive bootstrapping for the CKKS scheme in Protocol 3. The goal is as above, for multiple servers to collectively refresh an exhausted ciphertext. This protocol is based on the interactive bootstrapping protocol in [43, 48] and the security proof for threshold FHE in [3].

Consider k servers, $S_0, \ldots, S_{k-1}$. The collective secret key sk, which is associated to the collective public key pk, is additively secret-shared among the servers so that each server has access to its own secret key share s_i and the public key pk.

Let λ denote the statistical security level. In this protocol, we will make use of statistical masks, sampled uniformly from R_P, where $P \geq |\mathsf{m}|2^\lambda$, for the message sizes $|\mathsf{m}|$ supported. Adding such a mask to a message m ensures λ bits of statistical security. For correctness, the ciphertext modulus at which the interactive bootstrapping is initiated has to satisfy $q > kP$, in order to ensure the sum of masks does not overflow. Moreover, the ciphertext modulus q should also ensure that correct decryption of the masked value is possible. Finally, we specify χ_{DG} the Gaussian distribution over R with $\sigma = 3.19$ and maximum bound $\lfloor 6\sigma \rfloor$.

Protocol 3 Interactive Multiparty Bootstrapping Protocol $\Pi_{\mathcal{MIB}}$

Input: Each party has a secret key share $s_i \in R$, a common random polynomial $a \leftarrow R_Q$, and a ciphertext $\mathsf{ct}' = (a', b') \in R_q^2$ encrypted under a pk (with secret key $\mathsf{sk} = \sum_{i=0}^{k-1} s_i$) encrypting m.

Output: A fresh encryption $\mathsf{ct} \in R_Q^2$ under a pk, encrypting m.

```
1: for server $S_i$, $i = 0, \ldots, k-1$ do        ▷ Each server computes the following:
2:     $M_i \leftarrow R_P$, $e_{1,i} \leftarrow \chi_{DG}$
3:     $h_{0,i} \leftarrow s_i a' + M_i$
4:     $h_{1,i} \leftarrow -s_i a - M_i + e_{1,i}$
5: end for
6: $h_0 \leftarrow \sum_{i=0}^{k-1} h_{0,i}$        ▷ Only lead server computes the following:
7: $h_1 \leftarrow \sum_{i=0}^{k-1} h_{1,i}$
8: $b \leftarrow b' + h_0 \mod q$
9: $b \leftarrow b + h_1 \mod Q$
10: return $\mathsf{ct} = (a, b) \in R_Q^2$ to all servers
```

Theorem 3. *The protocol $\Pi_{\mathcal{MIB}}$ in Protocol 3 is correct as long as $q \geq k|\mathsf{m}|2^\lambda + 2|m|$.*

Proof. We have correctness if ct decrypts to the same message m as ct': $a'(\sum_{i=0}^{k-1} s_i) + b' = \mathsf{m} + e'$

(mod q). Let us compute:

$$\begin{aligned}
a\sum_{i=0}^{k-1} s_i + b &= a\sum_{i=0}^{k-1} s_i + (b' + h_1 \pmod q) + h_0 \pmod Q \\
&= a\sum_{i=0}^{k-1} s_i + \left(b' + a'\sum_{i=0}^{k-1} s_i + \sum_{i=0}^{k-1} M_i \pmod q\right) - a\sum_{i=0}^{k-1} s_i - \sum_{i=0}^{k-1} M_i + \sum_{i=0}^{k-1} e_{1,i} \pmod Q \\
&= \left(m + e' + \sum_{i=0}^{k-1} M_i \pmod q\right) - \sum_{i=0}^{k-1} M_i + \sum_{i=0}^{k-1} e_{1,i} \pmod Q \\
&\overset{(*)}{=} \left(\mathsf{m} + e' \pmod q\right) + \sum_{i=0}^{k-1} e_{1,i} \pmod Q = \mathsf{m} + e' + \sum_{i=0}^{k-1} e_{1,i} \pmod Q,
\end{aligned}$$

where the equality $(*)$ happens under the assumption there is no overflow over q of $\sum_{i=0}^{n-1} M_i$ and of the parenthesis, which holds under the assumption in the theorem statement. □

The ideal functionality of the multiparty interactive protocol takes as input the exhausted ciphertext and all the servers' secret key shares, computes the joint secret key and decrypts the ciphertexts, then computes a fresh encryption of the obtained message and outputs it to all servers.

Theorem 4. *The protocol $\Pi_{\mathcal{MIB}}$ in Protocol 3 is secure in the semi-honest model assuming that RLWE ciphertexts are pseudorandom under secret-shared secret keys, given all but one share, achieving the ideal functionality described above.*

Proof sketch. The arguments for security follow the proof in [48] on the backbone of [43]. The only difference is that the protocol in [48] samples an additional Gaussian noise to add to $h_{0,i}$. However, it is sufficient to add an extra Gaussian noise apart from masking in only one of the partial decryption terms, and still guarantee that no information is leaked by processing h_0 and h_1. Masking $h_{0,i}$ by a large uniformly random mask M_i still guarantees statistical security of $s_i a'$ and their sum, and security of $s_i a$ and their sum can still be argued via multi-secret RLWE. □

Practical considerations. In the multi-precision representation, the minimum ciphertext modulus can be chosen from the statement of Theorem 3, and for 2 servers and $\lambda = 128$ bits, it results in requiring 180 bits for plaintexts up to 50 bits. However, in the RNS representation, which is the preferred representation in practice, the minimum ciphertext modulus needs to be chosen as follows. In the case of 64-bit native integer size (where one RNS limb can be at most 60 bits), the mask modulus P has to have three limbs, and we need to save one limb for correct decryption. Therefore, RNS-friendliness trade-offs speed for requiring three extra limbs, whereas the multi-precision representation corresponds to only two extra limbs for the interactive bootstrapping. In the 128-bit RNS version, only two extra limbs are required as well.

The multiparty interactive bootstrapping protocol is also amenable to the variant of server–multi-client where there is only one lead server starting with the ciphertext and which aggregates the partial decryptions to obtain the refreshed ciphertext, without outputting it to the others.

F Private Join-and-Compute

In this section, we describe a multiparty private join-and-compute protocol that allows a set of *Data Owners* (DOs) to compute the join (intersection) of their data sets and perform a computation on the result, with the help of a *Computation Party* (CP). Each DO's data set is comprised of identifiers and features records. We emphasize that the join is performed only over the identifiers, i.e., the goal is to obtain the list of identifiers that are in all the data sets, and to perform computations over the features records corresponding to the identifiers in the intersection.

We will describe different protocol versions for the join and the compute parts of the protocol that trade privacy at the DOs with the overall communication and computation efficiency. For the join part, the first protocol we describe returns the identifiers in the intersection to the DOs. The second protocol hides the intersection indices from the DOs. The former leads to a more efficient compute part but reveals more information to the DOs while the latter has more privacy but requires more homomorphic computations by the CP in the compute part. Both protocol versions achieve semi-honest security (Definition 2) when all but one DOs collude; the first version, where the DOs learn the intersection, can also tolerate all but one DO and the CP collude. We note that here, the same party cannot be both a DO and a CP.

Our protocols extend a solution proposed by Ion et al. [33], where there are only two DOs and one of them also serves as the CP, only one of the two DOs has data to be computed upon after the intersection of identifiers, and the only supported computation is encrypted addition. As in [33], we target a simple protocol, based on well-studied cryptographic primitives, that is appealing to be used by both the DOs and the CP.

We assume the same record identification scheme is used for all the Data Owners, i.e., each common record is uniquely described by the same identifier. To conceal the private data, the records get randomly shuffled and random identifiers get inserted. The join is performed based on exact matches and uses *commutative ciphers*. A deterministic commutative cipher based on an elliptic curve instantiation of the DDH problem is used to compute a common hash (encryption) for each record. The CP learns the intersection size (along with the sizes of the possible subset-intersections of the data sets). The subsequent computations performed at the CP will be performed over homomorphically encrypted records in the intersection and is independent of the private join protocol.

Commutative ciphers The join protocol is based on exponentiation in a finite (multiplicative) group G, where the Decisional Diffie-Hellman (DDH) problem is believed to be hard. The group is used to implement a *commutative cipher*, i.e., $E_s(g) = g^s$, a form of deterministic encryption such that

$$E_z(E_s(g)) = E_s(E_z(g))$$

for any two keys s, z and group element g. The keys $s \in \mathbb{Z}_p^*$ are integers that are invertible modulo the size of the group $p = |G|$. For simplicity of the analysis, we may assume p is prime, so that $s \in \{1, \ldots, p-1\}$. Encryption also requires a hash function $\mathsf{H} : X \to G$ from the set of (application dependent) identifier values to group elements, modeled as a random oracle.

For implementation, we use elliptic curve groups, such as Curve25519 [6], implemented in the OpenSSL library (`https://www.openssl.org/`). SHA-256 is used for the random oracle, and is run until the result lies on the curve (we do not consider timing attacks in this paper, but there are methods to avoid such attacks if desired).

Matrix DDH We will use the following extended version of DDH (Definition 3), which can be reduced to the standard DDH assumption (Definition 1).

Definition 3 (Matrix DDH)**.** *The (n,m)-DDH problem asks to distinguish between $[g^{x_i y_j} : i,j] \in G^{n\times m}$ for $x_0 = y_0 = 1$, $x_i, y_j \sim \mathcal{U}(\mathbb{Z}_p^*)$ and a uniformly random matrix over $G^{n\times m}$.*

The standard DDH problem is just a special case: $(2,2)$-DDH. We claim that under DDH, the (n,m)-DDH problem is hard for any polynomially-bounded n,m.

Looking ahead, when analyzing security against a corrupt CP, it is in fact enough to use $(2,n)$-DDH. The $(2,n)$-DDH problem has a very simple proof under the DDH assumption, so we present this result separately as it also helps build intuition for the general result.

Lemma 1. *DDH implies $(2,n)$-DDH for any $n > 2$.*

Proof. For any integer matrix $\mathbf{X}$, we write $g^{\mathbf{X}}$ for the matrix of group elements $[g^{x_{i,j}} : i,j]$. Let $g^{\mathbf{X}}$ be an input to the DDH problem, with $\mathbf{X} \in (\mathbb{Z}_p^*)^{2\times 2}$. We can build a larger matrix $Y \in (\mathbb{Z}_p^*)^{2\times n}$ as follows. Each column of $\mathbf{Y}$ is computed as $\mathbf{X}\cdot\mathbf{z}_i$, where $\mathbf{z}_i \in (\mathbb{Z}_p^*)^2$ is a uniformly random vector. It is easy to see that $g^{\mathbf{Y}}$ can be easily computed from $g^{\mathbf{X}}$ by performing the operations in the exponent. Moreover, if $g^{\mathbf{X}}$ is a DDH matrix, then $g^{\mathbf{Y}}$ is a $(2,n)$-DDH matrix. Similarly, assuming $\mathbf{X}$ is invertible (which is true with high probability when $\mathbf{X}$ is uniformly random) then $\mathbf{Y}$ is also uniformly random. □

The following is a special case of Bresson et al.'s main result [10].

Lemma 2. *DDH implies (n,m)-DDH.*

Proof. We first prove that (n,n)-DDH implies $(n+1,n+1)$-DDH. Consider the matrix of exponents, $\mathbf{A}_n$, of an (n,n)-DDH instance: $\mathbf{A}_n = \mathbf{a}\mathbf{b}^t$ where $\mathbf{a} = (1, a_1, \ldots, a_{n-1})$ and $\mathbf{b} = (1, b_1, \ldots, b_{n-1})$ where $a_i, b_i \leftarrow \mathbb{Z}_p^*$. If we sample uniformly at random $\mathbf{x}, \mathbf{y} \leftarrow (\mathbb{Z}_p^*)^n$, then for $\mathbf{x}_n = \mathbf{A}_n\mathbf{x}$, $\mathbf{y}_n = \mathbf{A}_n^t\mathbf{y}$, and $a_{n+1} = \mathbf{y}^t\mathbf{A}_n\mathbf{x}$, $\mathbf{A}_{n+1} := \begin{pmatrix}\mathbf{A}_n & \mathbf{x}_n \\ \mathbf{y}_n^t & a_{n+1}\end{pmatrix}$ is the $(n+1,n+1)$-DDH distribution, since $\mathbf{A}_{n+1} = \begin{bmatrix}\mathbf{a} \\ \mathbf{y}^t\mathbf{a}\end{bmatrix}\begin{bmatrix}\mathbf{b}^t & \mathbf{b}^t\mathbf{x}\end{bmatrix}$. On the other hand, $\mathbf{x}_n$ and $\mathbf{y}_n$ are distributed uniformly at random as long as the matrix $\mathbf{A}_n$ is uniformly random and invertible (which happens with overwhelming probability). Then, we can replace a_{n+1} with a uniformly random exponent $a_{n+1} \leftarrow \mathbb{Z}_p^*$. The exact hybrids are as follows:

$$
\begin{aligned}
&\{\mathbf{A}_{n+1} | \mathbf{A}_n \leftarrow (n,n)\text{-DDH}, \mathbf{x} \leftarrow (\mathbb{Z}_p^*)^n, \mathbf{y} \leftarrow (\mathbb{Z}_p^*)^n, \mathbf{x}_n = \mathbf{A}_n\mathbf{x}, \mathbf{y}_n = \mathbf{A}_n^t\mathbf{y}, a_{n+1} = \mathbf{y}^t\mathbf{A}_n\mathbf{x}\} \approx_c \\
&\{\mathbf{A}_{n+1} | \mathbf{A}_n \leftarrow (\mathbb{Z}_p^*)^{n\times n}, \mathbf{x} \leftarrow (\mathbb{Z}_p^*)^n, \mathbf{y} \leftarrow (\mathbb{Z}_p^*)^n, \mathbf{x}_n = \mathbf{A}_n\mathbf{x}, \mathbf{y}_n = \mathbf{A}_n^t\mathbf{y}, a_{n+1} = \mathbf{y}^t\mathbf{A}_n\mathbf{x}\} \approx_s \\
&\{\mathbf{A}_{n+1} | \mathbf{A}_n \leftarrow (\mathbb{Z}_p^*)^{n\times n}, \mathbf{x} \leftarrow (\mathbb{Z}_p^*)^n, \mathbf{y} \leftarrow (\mathbb{Z}_p^*)^n, \mathbf{x}_n \leftarrow (\mathbb{Z}_p^*)^n, \mathbf{y}_n \leftarrow (\mathbb{Z}_p^*)^n, a_{n+1} = \mathbf{y}^t\mathbf{A}_n\mathbf{x}\} \approx_c \\
&\{\mathbf{A}_{n+1} | \mathbf{A}_n \leftarrow (\mathbb{Z}_p^*)^{n\times n}, \mathbf{x}_n \leftarrow (\mathbb{Z}_p^*)^n, \mathbf{y}_n \leftarrow (\mathbb{Z}_p^*)^n, a_{n+1} \leftarrow \mathbb{Z}_p^*\}.
\end{aligned}
$$

The first hybrid is from the pseudorandomness of (n,n)-DDH. The last hybrid is from DDH. For $m > n$, the same technique applies from Lemma 1's proof.

□

Protocol $\Pi_{\mathcal{PJ}}$ We first describe the flow of the protocol, sketched in Figure 4, and provide a concise description in Protocol 4. Each DO_i has a key $s_i \in \mathbb{Z}_p^*$, and a list of data set identifiers $X_i' \subset X$. As part of the pre-processing described in the main paper, each DO pads its set of record identifiers in the data set up to a size $n_{max} \geq \max\{|X_1'|, \ldots, |X_k'|\}$. We denote the padded data

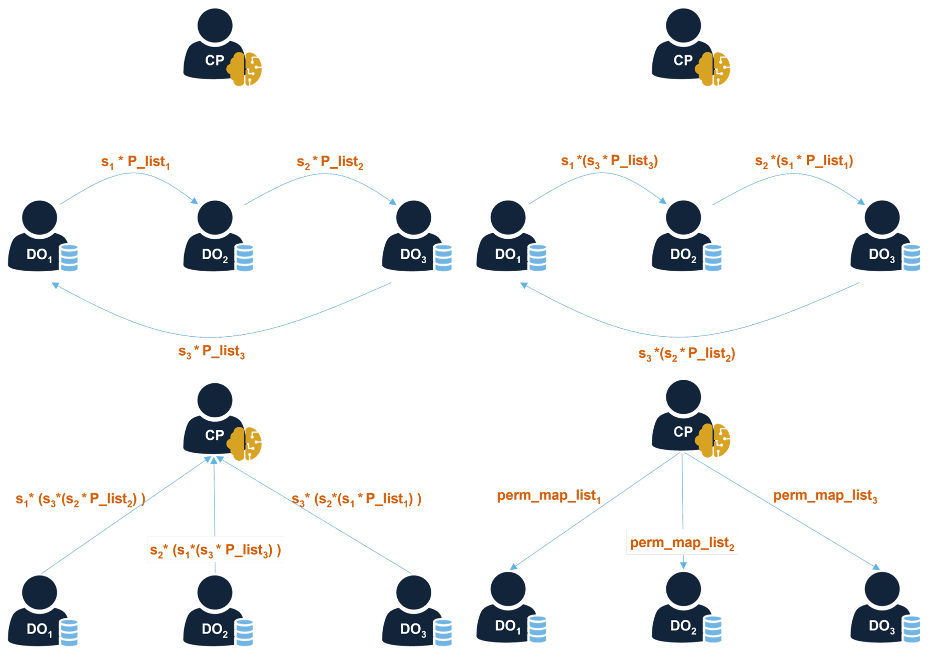

CP
s_1 * P_list$_1$
s_2 * P_list$_2$
s_3 * P_list$_3$
DO$_1$
DO$_2$
DO$_3$
CP
s_1 *(s_3 * P_list$_3$)
s_2 *(s_1 * P_list$_1$)
s_3 *(s_2 * P_list$_2$)
DO$_1$
DO$_2$
DO$_3$
CP
s_1* (s_3*(s_2 * P_list$_2$))
s_2* (s_1*(s_3 * P_list$_3$))
s_3* (s_2*(s_1 * P_list$_1$))
DO$_1$
DO$_2$
DO$_3$
CP
perm_map_list$_1$
perm_map_list$_2$
perm_map_list$_3$
DO$_1$
DO$_2$
DO$_3$

Figure 4: Diagrams of the steps of the private join protocol $\Pi_{\mathcal{PJ}}$ for $k = 3$ data owners (DO). Each DO randomly permutes their identifiers lists and stores this permutation as a secret. It also generates a random $s_i \in \mathbb{Z}_p^*$ as a secret exponent. In the figure, we use the multiplicative notation. DO_i hashes its padded identifier list X_i to the DDH group G and exponentiates the hashed result, entry-wise: $(\mathsf{H}(x')^{s_i} : x' \in S_i' = \pi_i(X_i)) \in G^{n_{max}}$, where π_i is the aforementioned private permutation. This corresponds to $s_i *$ P_list$_i$ in the diagram, for DO_i. Then, each party passes their exponentiated group elements to the next DO, which raises the received list to their secret exponent. Finally, the lists are sent to the Computation Party (CP) which computes the intersection and the corresponding indices in each permuted data set. The last diagram corresponds to step (vi*) in the description. There, we depict the CP sending back a permuted list of indices (corresponding to perm_map_list$_i$ for DO_i in the figure) that captures both the case where the CP sends back the exact intersection list and the case where the CP adds more random indices and scrambles the order to reduce the information leaked about the intersection records to the DOs.

set of DO_i as X_i, each of length n_{max}. The padding is done via hash functions with outputs in different domains, such that we do not introduce false intersecting identifiers (the values in the data set corresponding to the padded identifiers can be zeroes). The protocol proceeds as follows:

(i) Each $\mathsf{DO}_i, i \in \{1, \dots, k\}$, randomly permutes the set $S_i' = \pi_i(X_i)$, and hashes the elements of the permuted set to obtain a vector of random group elements:

$$\mathbf{m}'[i,i] = (\mathsf{H}(x') : x' \in S_i') \in G^{n_{max}}$$

(ii) During the execution of the protocol, the parties compute and transmit a collection of messages

$$\mathbf{m}[i,j] \in G^{n_{max}}$$

indexed by $i, j \in \{1, \dots, k\}$, and each consisting of vectors of n_{max} group elements.

(iii) Messages $\mathbf{m}[i,j]$ are transmitted in k rounds, with message $\mathbf{m}[i,j]$ sent in round $j-i \pmod{k}$ from DO_{j-1} to DO_j.

(iv) Each DO_i computes the messages as follows:

$$\mathbf{m}[j,i+1] = \mathbf{m}[j,i]^{s_i}$$

where the exponentiation is applied entry-wise. Message $\mathbf{m}[j,i+1]$ is computed by DO_i, who knows s_i and receives $\mathbf{m}[j,i]$ from DO_{i-1}. All DO_i indexes computed modulo k. As a special case, the first message $\mathbf{m}[i,i+1]$ is computed using the vector $\mathbf{m}'[i,i]$ locally computed by DO_i from its input. Also, the last message $\mathbf{m}[i,i]$ computed by DO_i is sent to CP rather than DO_{i+1}.

(v) CP receives a set of messages

$$\mathbf{m}[i,i] = \pi_i(\mathsf{H}(X_i))^s$$

containing the permuted identifier sets X_i encrypted under $s = \prod_i s_i$ the product of all keys. Since exponentiation is commutative and deterministic, this allows to compute the set intersection on the ciphertexts, and determine for each i a set of indexes J_i such that $X_i[\pi_i^{-1}(J_i)] = S = \bigcap_j X_j$.

(vi*) DO_i receives the indices J_i of its portion of the intersection from CP and computes its local output $X_i[\pi_i^{-1}(J_i)]$.

To clarify each of the DO's steps, in Protocol 4, we use an "input port," from which the DO reads the message to update, updates it, and writes it for the future round.

Protocol 4 Interactive protocol for private join $\Pi_{\mathcal{PJ}}$

```
Input: Each $\mathsf{DO}_i$'s input set of padded identifiers $X_i$.
Output: $\bigcap_i X_i$ to each $\mathsf{DO}_i$.
1: Each $\mathsf{DO}_i$ samples $s_i \leftarrow \mathbb{Z}_p^*$ and a permutation $\pi_i : [n_{max}] \to [n_{max}]$. It
   then computes $\mathbf{m}'[i,i] := (\mathsf{H}(x') : x' \in S_i' = \pi_i(X_i)) \in G^{n_{max}}$. Place
   $\mathbf{g}_{in} := \mathbf{m}'[i,i]$ to its own input port.
2: For $j = 1, \ldots, k$, each $\mathsf{DO}_i$:
     • Read in $\mathbf{g}_{in}$ from the input port.
     • Compute $\mathbf{g}' := \mathbf{g}_{in}^{s_i}$.
     • Send $\mathbf{g}'$ to $\mathsf{DO}_{(i+1)\%k}$'s input port if $j < k$ and send $\mathbf{g}'$ to $\mathsf{CP}$ otherwise.
3: $\mathsf{CP}$ gets $\pi_i(\mathsf{H}(X_i))^s$, $s = \prod_j s_j$, from $\mathsf{DO}_{i-1}$. Then, it computes the inter-
   section from these lists.
4*: $\mathsf{CP}$ sends the indices $J_i$ to $\mathsf{DO}_i$, which computes $\pi_i^{-1}(J_i)$.
```

After step (v) of the private join (line 3 in Protocol 4), the CP has the lists of indices J_i corresponding to the intersection under permutation π_i. The next step is for the CP to obtain the encrypted records corresponding to the intersection from the DOs. There are several options to get to this result, offering a compromise between computation and communication efficiency and the amount of information revealed to the Data Owners.

One option is for the CP to send back J_i to DO_i at the end of the private join protocol (as depicted in step (vi*) or line 4* in Protocol 4). Then, each DO_i applies its local π_i^{-1} to recover the intersection indices in its data set, encrypts only the records corresponding to the intersection using the joint public key of the threshold FHE scheme, and sends the ciphertexts to the CP. This solution corresponds to the minimum amount of communication from the DOs to the CP, as well as the minimum amount of computation at the CP for aligning the records in the intersection, but it reveals the intersection identifiers to the DOs.

At the other extreme, the CP does not send back the intersection indices J_i to the DO_i (the protocol ends after step (v) or line 3 in Protocol 4), and each DO_i encrypts their whole data set of records (permuted by π) and sends them to the CP, which needs to perform homomorphic computations to align the encrypted records in the intersection. The CP uses the lists $(J_i)_{i\in\{1,\dots,k\}}$ to multiplicatively mask the encrypted slots corresponding to the indices not in the intersection, in order to obtain the merged encrypted data set. This solution corresponds to the maximum amount of communication from the DOs to the CP and maximum amount of computation at the CP, but the DOs do not learn anything about the intersection.

In the implementation, we take a middle approach, which has the same communication as the latter version but substantially saves on computation, at the cost of some privacy leakage. The CP will send to each DO_i a (different) list K_i of size of n_{max}, containing the (permuted) indices in J_i but also random indices. Importantly, the positions of the indices in the intersection will be the same over all lists sent by the CP, such that they are already aligned in the final encrypted data sets. The DOs will apply the inverse of their local permutation they used in the commutative cipher phase, to reorder their local records according to the list received from the CP, then encrypt their data sets in this order. The advantage of this approach is that it saves substantial computation at the CP, i.e., the computation necessary to align the encrypted values. In more detail, when constructing the lists of indices for each DO, the CP will try to maximize the number of positions of the indices in the pairwise intersections $\pi_i(X_i) \cap \pi_j(X_j)$ that are on the same position in the list K_i for DO_i and list K_j for DO_j. When there are no collusions between the corrupted parties, this protocol retains the security guarantee that no party learns the identifiers in the intersection. When there are collusions, the corrupted parties might learn more than their pairwise intersection, and privacy could default to the case where the indices in the intersection are revealed (but never the indices that are not in the intersection or the values of the records).

In the following, we present the security proofs for the extreme cases of private join with respect to Definition 2. Then, we analyze the security of the intermediate version.

Security Proofs We start with the private join protocol that returns output to the data owners, i.e., line 4* is executed. The ideal functionality for the protocol is given in Functionality 5. Every party knows the size n_{max} of each DO's padded identifier lists. Define $w_S := |\bigcap_{j\in S} X_j|$. For k data sets, there are $2^k - 1$ such non-trivial intersections. The CP will learn this information, but nothing more about the indices in the intersection.

Functionality 5 Ideal Functionality $\mathcal{F}_{\mathcal{PJ}}$ for private join protocol $\Pi_{\mathcal{PJ}}$

```
Input: Each $\mathsf{DO}_i$'s input set of identifiers $X_i$.
Output: $(\bigcap_i X_i)$ to each $\mathsf{DO}_i$ and $((w_S)_{S\subseteq\{1,\ldots,k\}})$ to $\mathsf{CP}$.
1: Compute the intersection $\bigcap_i X_i$.
2: Compute all intersection sizes, $w_S$ for $S \subseteq \{1,\ldots,k\}$.
3: return $(\bigcap_i X_i)$ to each $\mathsf{DO}_i$ and $(w_S)_{S\subseteq\{1,\ldots,k\}}$ to $\mathsf{CP}$.
```

Theorem 5 shows semi-honest security against a corrupted CP. The simulator in the proof is exponential-time in the number of parties. Therefore, the protocol is secure when $k = O(\log(\kappa))$. This is not a limition for our application since the number of parties we consider is small.

Theorem 5. *Protocol 4 with output returned to the* DO*s is secure against a corrupted* CP*.*

Proof. The input, output and view of the CP are:

$\mathsf{Input}^{\Pi}_{\mathsf{CP}} : \emptyset$.
$\mathsf{Output}^{\Pi}_{\mathsf{CP}} : (w_S)_{S\subseteq\{1,\ldots,k\}}$.
$\mathsf{View}^{\Pi}_{\mathsf{CP}} : \big((\mathbf{m}[j,j], J_j)_{j\in\{1,\ldots,k\}}, (w_S)_{S\subseteq\{1,\ldots,k\}}\big)$.

The view of CP consists of a collection of messages

$$\mathbf{m}[j,j] = \mathsf{H}(\pi_j(X_j))^s = \pi_j(\mathsf{H}(X_j)^s)$$

where $s = \prod_j s_j$ is the product of all secret keys, and π_j is a random permutation.

$\underline{\mathsf{Sim}_{\mathsf{CP}}((w_S)_{S\subseteq\{1,\ldots,k\}})}$:

1. For $S = \{1,\ldots,k\}$, set $z_S \leftarrow w_S$.
2. For $i = k-1 : 1$, for each $S \subseteq \{1,\ldots,k\}$ of size $|S| = i$, do:
   set $z_S \leftarrow w_S - \sum_{T\neq\emptyset}^{\bar{S}} z_{S\cup T}$, where $\bar{S}$ is the complement of S in $\{1,\ldots,k\}$.
3. For each $S \subseteq \{1,\ldots,k\}$, sample z_S uniformly random group elements without replacement and include them in a set B_j, for each $j \in S$.
4. For each $j \in \{1,\ldots,k\}$, set $\hat{\mathbf{m}}[j,j] = \hat{\pi}_j(B_j)$, where $\hat{\pi}_j$ is a uniformly random permutation.
5. Compute $\hat{J}_j$ according to the intersection indices in B_j and $\hat{\pi}_j$ and output $\Big((\hat{\mathbf{m}}[j,j], \hat{J}_j)_{j\in\{1,\ldots,k\}}, (w_S)_{S\subseteq\{1,\ldots,k\}}\Big)$.

Note that the simulator computes the size $z_S = |Z_S|$ of the sets

$$Z_S = \bigcap_{i\in S} X_i \setminus \bigcup_{j\notin S} X_j.$$

The sets Z_S represent the elements that belong precisely to each subset S of data owners. Moreover, the sets Z_S are disjoint and their union equals $\bigcup_j X_j$. The sampling in step 4 happens without replacement across elements of S and across the values of S. Notice that by construction, the final size of the (multi)sets B_j is $|B_j| = n_{max}$, and the construction in step 4 ensures that the intersection sizes are satisfied.

We want to prove indistinguishability between:

$$\left((\hat{\mathbf{m}}[j,j], \hat{J}_j)_j, \cap_j X_j, (w_S)_{S\subseteq\{1,\ldots,k\}}\right) \approx_c \left((\mathbf{m}[j,j], J_j)_j, \cap_j X_j, (w_S)_{S\subseteq\{1,\ldots,k\}}\right).$$

In order to prove that the simulated view is indistinguishable from a real execution, we consider the following experiment. The experiment uses a list of pairs (g_i, h_i) where the g_i's are chosen uniformly at random, and the h_i's are either set to $h_i = g_i^s$ for a fixed random value s, or they are chosen uniformly, independently at random. Notice that, by Lemma 2, these two distributions are computationally indistinguishable under the DDH assumption on G.

Whenever an adversary queries the hash function, $\mathsf{H}(x_i)$ for $x_i \in X_i$, search in the lookup table to see if x_i was queried before. If so, return the value in the table, if not, sample a uniformly random group element $g_i \leftarrow G$, store it in the table indexed by i and return $\mathsf{H}(x_i) = g_i$. It is easy to see that when $h_i = g_i^s$, then this has the same distribution as the real execution of the protocol. On the other hand, when the h_is are chosen independently at random, the final output has the same distribution as the view output by the simulator. Therefore, by Lemma 2, the real view and the simulated view are computationally indistinguishable. □

Theorem 6. *Protocol 4 with output returned to the* DO*s is secure against a corrupted* DO *assuming* $(n \times m)$*-DDH.*

Proof. Without loss of generality, let DO_k be the corrupted party. Let L_k denote the indices of the intersection $\bigcap_j X_j$ within X_k. All indices are modulo k.

$\mathsf{Input}^{\Pi}_{\mathsf{DO}_k} : X_k$.
$\mathsf{Output}^{\Pi}_{\mathsf{DO}_k} : \bigcap_j X_j$.
$\mathsf{View}^{\Pi}_{\mathsf{DO}_k} : \left(X_k, \mathsf{coins}_{\mathsf{DO}_k}, \bigcap_j X_j; \mathbf{m}[k-1,k], \mathbf{m}[k-2,k], \ldots, \mathbf{m}[1,k], \pi_k(L_k)\right)$.

$\underline{\mathsf{Sim}_{\mathsf{DO}_k}(X_k, (\bigcap_j X_j))}$:

1. Sample $\widehat{\mathsf{coins}}_{\mathsf{DO}_k} = (\hat{s}_k, \hat{\pi}_k)$, where $\hat{s}_k \leftarrow \mathbb{Z}_p^*$ is a random element and $\hat{\pi}_k$ a random permutation.
2. For round $l \in \{1, \ldots, k-1\}$, sample n_{max} random elements in G, $\hat{\mathbf{g}}_l \leftarrow G^{n_{max}}$ uniformly at random (corresponding to $\mathbf{m}[k-l,k]$ in the real protocol).
3. Compute $\hat{\pi}_k(L_k)$ using $\bigcap_j X_j$ and X_k.
4. Output $\left(X_k, \widehat{\mathsf{coins}}_{\mathsf{DO}_k}, \bigcap_j X_j; \hat{\mathbf{g}}_1, \hat{\mathbf{g}}_2, \ldots, \hat{\mathbf{g}}_{k-1}, \hat{\pi}_k(L_k)\right)$.

Whenever the adversary queries the hash function, $\mathsf{H}(y_j)$, search the lookup table to see if y_j was queried before. If so, return the value in the table. If not, sample a uniformly random group element $g_{y_j} \leftarrow G$, store it in the table indexed by y_j, and return g_{y_j}.

Note that the real output and ideal output of CP, $(w_S)_S$, are the same given the same input data sets, so as long as the simulated view is consistent with the inputs X_i, it will also be consistent with $(w_S)_S$. Now we show that the simulated view is computationally indistinguishable from the real view assuming $(n \times m)$-DDH. For simplicity, first assume that any element in a pairwise intersection is in all the sets: $x \in X_i \cap X_j \implies x \in \bigcap_j X_j$. Fix N elements in the intersection $x_i \in \bigcap_j X_j$ for $i \in \{1, 2, \ldots, k\}$. Then, let $g := \mathsf{H}(x_1)$. There exists an r_i such that $g^{r_i} = \mathsf{H}(x_i)$ for $i \geq 2$. Let $g_i := \mathsf{H}(x_i)$. The adversary gets to see the following matrix of group elements in the real protocol up to permutations:

$$\begin{pmatrix} g & g^{s_{k-1}} & g^{s_{k-1}s_{k-2}} & \cdots & g^{s_{k-1}s_{k-2}\cdots s_1} \\ g^{r_2} & g^{r_2 s_{k-1}} & g^{r_2 s_{k-1}s_{k-2}} & \cdots & g^{r_2 s_{k-1}s_{k-2}\cdots s_1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ g^{r_N} & g^{r_N s_{k-1}} & g^{r_N s_{k-1}s_{k-2}} & \cdots & g^{r_N s_{k-1}s_{k-2}\cdots s_1} \end{pmatrix}$$

which is exactly the $(N \times k)$-DDH distribution, up to permutations. Column $i = 0, 1, 2, \ldots, k-1$ in the matrix corresponds to what elements DO_k sees in round i that are in the intersection. The round $i = 0$ corresponds to DO_k hashing these elements.

All elements not in the intersection are truly uniformly random since we are hashing onto the group in the random oracle model and exponentiation is an invertible function on $G \setminus \{e\}$ where e is the group identity. At the same time, note that the simulated messages $(\hat{\mathbf{g}}_1, \ldots, \hat{\mathbf{g}}_{k-1})$ form a matrix uniformly distributed, hence, by the DDH problem (Lemma 2), the ideal and real views are computationally indistinguishable.

In the case where some pairwise-non-distinct elements are not in $\bigcap_j X_j$, then this is the above distribution with entries missing. $\square$

Theorem 7. *Protocol 4 with output returned to the DOs is secure against up to $k-1$ colluding DOs assuming $(n \times m)$-DDH.*

Proof. We write the simulator for the case where there are $k-1$ colluding parties and argue the more general case. Without loss of generality, assume DO_1 is the only honest party. We abbreviate $\{\mathsf{DO}_i\}_{i \geq 2}$ (and $\{X_i\}_{i \geq 2}$) with $\{\mathsf{DO}_i\}$ (and $\{X_i\}$) below. Let L_i denote the indices of the intersection $\bigcap_j X_j$ within X_i.

$\mathsf{Input}^{\Pi}_{\{\mathsf{DO}_i\}} : \{X_i\}$.
$\mathsf{Output}^{\Pi}_{\{\mathsf{DO}_i\}} : \bigcap_j X_j$.
$\mathsf{View}^{\Pi}_{\{\mathsf{DO}_i\}} : \Big(\{X_i\}, \{\mathsf{coins}_{\mathsf{DO}_i}\}, \bigcap_j X_j; (\mathbf{m}[k-i, k])_{i=1}^{k-1}, (\mathbf{m}[k-1-i, k-1])_{i=1}^{k-1}, \ldots, (\mathbf{m}[2-i, 2])_{i=1}^{k-1}, \pi_i(L_i)\Big)$.

$\underline{\mathsf{Sim}_{\mathsf{DO}_1}(X_1, (\bigcap_j X_j))}$: For $i \geq 2$ repeat the following:

1. Sample $\{\widehat{\mathsf{coins}}_{\mathsf{DO}_i}\} = \{(\hat{s}_i, \hat{\pi}_i)\}$, where $\hat{s}_i \leftarrow \mathbb{Z}_p^*$ is a random element and $\hat{\pi}_i$ is a random permutation.
2. For round $l \in \{1, \ldots, k-1\}$, sample n_{max} random elements in G, $\hat{\mathbf{g}}^i_l \leftarrow G^{n_{max}}$ uniformly at random (corresponding to $\mathbf{m}[i-l, i]$ in the real protocol).
3. Compute $\hat{\pi}_i(L_i)$ using $\bigcap_j X_j$ and X_i.
4. Append to the output $\Big(X_i, \widehat{\mathsf{coins}}_{\mathsf{DO}_i}, \bigcap_j X_j; \hat{\mathbf{g}}^i_1, \ldots, \hat{\mathbf{g}}^i_{i-1}, \hat{\mathbf{g}}^i_{i+1}, \ldots, \hat{\mathbf{g}}^i_k, \hat{\pi}_i(L_i)\Big)$.

Whenever the adversary queries the hash function, $\mathsf{H}(y_j)$, search the lookup table to see if y_j was queried before. If so, return the value in the table. If not, sample a uniformly random group element $g_{y_j} \leftarrow G$, store it in the table indexed by y_j, and return g_{y_j}.

When the only honest party is DO_1, consider the set $Y = X_1 \bigcap (\bigcup_{i\geq 2} X_i)$. Fix some y_0 in this set which hashes to $g := \mathsf{H}(y_0)$. Then, all other y_i for $i \geq 1$ in the set hash to $g^{r_i} := \mathsf{H}(y_i)$ and the view of the corrupted parties is some permutation of the matrix:

$$\begin{pmatrix} g & g^{s_1} \\ g^{r_2} & g^{r_2 s_1} \\ g^{r_3} & g^{r_3 s_1} \\ \vdots & \vdots \\ g^{r_M} & g^{r_M s_1} \end{pmatrix}$$

where $M = |X_1 \bigcap (\bigcup_{i\geq 2} X_i)|$. This is the $(2 \times M)$-DDH distribution. The remaining transcript is made up of uniformly random elements in G.

Now, assume there are $l \leq k-1$ honest parties and they are $X_1, X_2, \ldots, X_l$. Consider the set of elements in the union of the corrupt parties intersect with the union of the non-corrupt parties: $(\bigcup_{j\leq l} X_j) \bigcap (\bigcup_{i>l} X_i)$. Further, assume that if $x \in (\bigcup_{j\leq l} X_j) \bigcap (\bigcup_{i>l} X_i)$, then it is in each honest X_j for simplicity. Then, the view of the corrupted parties is

$$\begin{pmatrix} g & g^{s_l} & g^{s_l s_{l-1}} & \ldots & g^{s_l s_{l-1} \cdots s_1} \\ g^{r_2} & g^{r_2 s_l} & g^{r_2 s_l s_{l-1}} & \ldots & g^{r_2 s_l s_{l-1} \cdots s_1} \\ \vdots & \vdots & \vdots & & \vdots \\ g^{r_M} & g^{r_M s_l} & g^{r_M s_l s_{l-1}} & \ldots & g^{r_M s_l s_{l-1} \cdots s_1} \end{pmatrix}$$

This is the $(M \times (l+1))$-DDH distribution. The general case for the intersections has elements of this matrix missing. Further, the general case of l honest parties is the same since the cipher is commutative and exponentiation is an invertible function on G. □

Theorem 8. *Protocol 4 with output returned to the DOs is secure against up to $k-2$ colluding DOs and the CP assuming $(n_{max} \times (k+1))$-DDH.*

Proof. We start with the case where only one Data Owner, without loss of generality DO_k, colludes with the CP. Let J_k be the set of indices of the intersection sent by the CP to DO_k. Simulation is going to be backwards; first we construct the view of CP, then we construct the messages seen between the DO's with the correct dependency in the exponent with the CP's view. The view of this coalition is

$\mathsf{View}^{\Pi}_{\mathsf{CP},\mathsf{DO}_k} : \Big(X_k, \mathsf{coins}_{\mathsf{DO}_k}, \bigcap_j X_j; \mathbf{m}[k-1,k], \mathbf{m}[k-2,k], \ldots, \mathbf{m}[1,k], (\mathbf{m}[j,j], J_j)_{j\in\{1,\ldots,k\}}\Big)$.

The simulator for this coalitions is:

$\underline{\mathsf{Sim}_{\mathsf{CP},\mathsf{DO}_k}(X_k, \bigcap_j X_j, (w_S)_{S\subseteq\{1,\ldots,k\}})}$:

1. Sample $\widehat{\mathsf{coins}}_{\mathsf{DO}_k} = (\hat{s}_k, \hat{\pi}_k)$, where $\hat{s}_k \leftarrow \mathbb{Z}_p^*$ is a random element and $\hat{\pi}_k$ a random permutation.

2. For $S = \{1, \ldots, k\}$, set $z_S \leftarrow w_S$.

3. For $i = k-1 : 1$, for each $S \subseteq \{1, \ldots, k\}$ of size $|S| = i$, do:

   set $z_S \leftarrow w_S - \sum_{T \neq \emptyset}^{\bar{S}} z_{S \cup T}$, where $\bar{S}$ is the complement of S in $\{1, \ldots, k\}$.

4. For each $S \subseteq \{1, \ldots, k\}$, sample z_S uniformly random group elements without replacement and include them in a set B_j, for each $\{1, \ldots, k\}$.

5. For each $j \in \{1, \ldots, k-1\}$, set $\hat{\mathbf{m}}[j, j] = \hat{\pi}_j(B_j)$, where $\hat{\pi}_j$ is a uniformly random permutation. Set $\hat{\mathbf{m}}[k, k] = \hat{\pi}_k(B_k)$.

6. Compute $\hat{J}_j$ according to the intersection indices in B_j and $\hat{\pi}_j$ and output $(\hat{\mathbf{m}}[j, j], \hat{J}_j)_{j \in \{1, \ldots, k\}}$.

7. Compute $\hat{\mathbf{m}}[1, k] = \hat{\mathbf{m}}[k, k]^{\hat{s}_k^{-1}}$, then $\hat{\mathbf{m}}[2, k], \ldots, \hat{\mathbf{m}}[k-1, k]$ as uniformly random lists of group elements $\hat{\mathbf{g}}_l \leftarrow G^{n_{max}}$ for $l \geq 2$.

8. Output $\left(X_k, \widehat{\mathsf{coins}}_{\mathsf{DO}_k}, \bigcap_j X_j; \hat{\mathbf{m}}[k-1, k], \hat{\mathbf{m}}[k-2, k], \ldots, \hat{\mathbf{m}}[1, k], (\hat{\mathbf{m}}[j, j], \hat{J}_j)_{j \in \{1, \ldots, k\}}\right)$.

For simplicity, first assume that any element in X_k and another X_j is in the intersection ($x \in X_k \cap X_j \implies x \in \bigcap X_j$) and that there are $M \leq n_{max}$ of these elements. The view of the corrupted parties CP and DO_k in the real protocol in this simplified case includes the following matrix:

$$\begin{pmatrix} g & g^{s_{k-1}} & g^{s_{k-1}s_{k-2}} & \ldots & g^{s_{k-1}s_{k-2}\cdots s_1} & g^{s} \\ g^{r_2} & g^{r_2 s_{k-1}} & g^{r_2 s_{k-1}s_{k-2}} & \ldots & g^{r_2 s_{k-1}s_{k-2}\cdots s_1} & g^{r_2 s} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ g^{r_M} & g^{r_M s_{k-1}} & g^{r_M s_{k-1}s_{k-2}} & \ldots & g^{r_M s_{k-1}s_{k-2}\cdots s_1} & g^{r_M s} \end{pmatrix}.$$

Note that this is the same matrix in the proof of Theorem 6 plus an extra column which is its predecessor raised to the corrupt party's secret, s_k. Therefore, the view of DO_k is indistinguishable from random assuming $(M \times k)$-DDH and the view of the corrupt CP is indistinguishable from random besides on the intersections of size z_S which match. Assuming matrix DDH, the view is indistinguishable from the real view of the corrupted parties in the protocol.

Second, assume that there is only one honest party, DO_1. We construct the view of CP as above. The messages are pseudorandom conditioned on relations in the exponents with the messages seen by the CP. We align these dependencies as follows: first, note that $\hat{\mathbf{m}}[2, 2] = \hat{\mathbf{m}}[3, 2]^{\hat{s}_2}$ where $\hat{\mathbf{m}}[2, 2]$ is sent to the CP by DO_2 and DO_1 sends $\hat{\mathbf{m}}[3, 2]$ to DO_2 in round $2 - 3 \mod k = k - 1$. Next, see that $\hat{\mathbf{m}}[3, 3] = \hat{\mathbf{m}}[4, 2]^{\hat{s}_2 \hat{s}_3}$ where $\hat{\mathbf{m}}[4, 2]$ is sent from DO_1 to DO_2 in round $2 - 4 \mod k = k - 2$. Continuing this way, we see that $\hat{\mathbf{m}}[\alpha, \alpha] = \hat{\mathbf{m}}[\alpha + 1, 2]^{\hat{s}_2 \cdots \hat{s}_\alpha}$ where $\hat{\mathbf{m}}[\alpha + 1, 2]$ is the message sent from DO_1 to DO_2 in round $k - (\alpha - 1)$.

In other words, we set

$$\begin{aligned} \hat{\mathbf{m}}[3, 2] &= \hat{\mathbf{m}}[2, 2]^{1/\hat{s}_2} \\ \hat{\mathbf{m}}[4, 2] &= \hat{\mathbf{m}}[3, 3]^{1/(\hat{s}_2 \hat{s}_3)} \\ &\vdots \\ \hat{\mathbf{m}}[\alpha + 1, 2] &= \hat{\mathbf{m}}[\alpha, \alpha]^{1/(\hat{s}_2 \cdots \hat{s}_\alpha)} \\ &\vdots \\ \hat{\mathbf{m}}[1, 2] &= \hat{\mathbf{m}}[k, k]^{1/(\hat{s}_2 \cdots \hat{s}_k)} \end{aligned}$$

and the rest of DO_1's messages as random elements in $G^{n_{max}}$. The proof follows as above (the DDH matrix has two rows).

Finally, consider the general case where the honest parties are $\mathsf{DO}_1, \ldots, \mathsf{DO}_{l-1}$ without loss of generality. Just as before, we construct the CP's view as in the simulator above.

$$\begin{aligned}
\hat{\mathbf{m}}[l+1, l] &= \hat{\mathbf{m}}[l, l]^{1/\hat{s}_l} \\
\hat{\mathbf{m}}[l+2, l] &= \hat{\mathbf{m}}[l+1, l+1]^{1/(\hat{s}_l \hat{s}_{l+1})} \\
&\vdots \\
\hat{\mathbf{m}}[1, l] &= \hat{\mathbf{m}}[k, k]^{1/(\hat{s}_l \cdots \hat{s}_k)}.
\end{aligned}$$

The rest proof follows as above from matrix DDH. □

We now analyze the security of the private join protocol in Protocol 4 which does not return output to the DOs (line 4* is not executed). The ideal functionality changes from the one in Protocol 5 only by not returning $\bigcap_i X_i$ to the DOs. In the security proofs, not having the intersection as output means that neither the simulators for the DOs nor the distinguisher that has to distinguish between the ideal distribution and the joint distribution have access to $\bigcap_i X_i$ (since a coalition of all the Data Owners is not allowed).

We note that in this case, we do not allow a coalition between the CP and any DO, since it would reveal the intersection indices to this coalition. In more detail, when DO_k and CP collude, this coalition can look at the permuted indices which collide under the DDH encodings and the set X_k to compute the intersection. This would be more than what the ideal functionality would allow: only the CP gets the $\{w_S\}$ indices and the DOs get nothing. Hence, Theorem 8's analog in this case does not hold.

Theorem 9. *Protocol 4 without output to the DOs is secure against a corrupted CP.*

The proof of Theorem 9 follows immediately from the proof of Theorem 5, where the simulator $\mathsf{Sim}_{\mathsf{CP}}$ uses the output list of indices to construct the sets B_i in line 3, and line 5 is not executed.

Theorem 10. *Protocol 4 without output to the DOs is secure against a corrupted DO assuming* $(n \times m)$*-DDH.*

The proof of Theorem 10 follows immediately from the proof of Theorem 6, where the simulator $\mathsf{Sim}_{\mathsf{DO}_k}(X_k)$ is the same as the simulator in the proof of Theorem 6 but without performing step 3, and the argument for security follows the same as before by the generalized DDH problem.

Theorem 11. *Protocol 4 without output to the DOs is secure against up to* $k-1$ *colluding DOs assuming* $(n \times m)$*-DDH.*

Theorem 11's proof follows immediately from Theorem 7's proof since the proof is the same without the simulator's third step (computing $\hat{\pi}_i(L_i)$).

For the intermediate version, the security against a corrupted CP or against a corrupted DO (or when multiple DOs do not collude with each other) is the same as in Theorem 9 and Theorem 10. Since it only makes sense to tolerate up to $k-1$ colluding DOs (otherwise they trivially obtain the intersection), the implemented protocol is fully secure in the case of 2 DOs.

However, for $k > 2$ and colluding DOs, there is no algorithm for constructing k lists of indices at the CP that guarantees that, for any size of the input data sets n_{max} and any size of the intersection $|\bigcap_i X_i|$, the lists of indices sent to the DOs will be such that all elements in every pairwise intersection will be on the same positions. This means that a set of colluding DOs could potentially learn more than allowed (i.e., what they can obtain by comparing their local data sets) about the identifiers in the intersection, by comparing the intersections of their local identifiers with the indices that are on the same positions in the received lists. Quantifying the size of the leaked set is data-dependent and will be a function of the size of the intersection set, the size and the distribution of pairwise intersections at the DOs. In other words, security against colluding DOs can in the worst case default to Theorem 7, in the best case achieve Theorem 11, and in practice, be in-between.

Encrypted Computations The steps of the private compute part executed after the private join are sketched in Figure 5. After the private join, the data owners encrypt (some of) their features records using threshold FHE and send them to the CP, which might need to first perform computations to obtain the ciphertexts of the joined data, before proceeding with the rest of the computation. In the first case where the DO_i receives the indices J_i from CP, it encrypts via threshold FHE the records from its database with the indices corresponding to $\pi_i^{-1}(J_i)$ and sends the ciphertext(s) to the CP. The ciphertext(s) the CP collects are already aligned. In the second case where the DO_i does not receive an output from the CP, it encrypts via FHE all of its features records permuted by π_i. The CP masks (multiplicatively via zeros-ones masks) the received ciphertext(s) such that only the slots in the intersection, indicated by J_i, are multiplied by ones, and the rest of the slots are multiplied by zeros. A further post-processing step is required in order to align the ciphertexts coming from different DOs, and the CP achieves this by using homomorphic rotations corresponding to the indices in $(J_j)_{j\in\{1,\dots,k\}}$, homomorphic additions and more homomorphic multiplicative masks. In the implemented version, the only difference from the latter case is that the DOs send the encrypted records with indices $\pi_i^{-1}(K_i)$, where K_i is the list of indices received from the CP, and that the CP only needs to perform the multiplicative masks to zero out the records that are not in the intersection.

Once the aligned ciphertexts are obtained, the CP proceeds with the desired computations. When the depth of the computations is larger than the allowed depth by the parameter, the parties will perform interactive bootstrapping: multiparty bootstrapping when the number of data owners $k > 1$ and two-party bootstrapping when there is only one data owner. Finally, the CP and DOs perform joint decryptions on the result ciphertext(s) at the CP such that the cleartext result is obtained.

Note that in neither version of the private compute protocol, the CP does not learn any more information about the data sets of the DOs or about their intersection, since it purely post-processes encrypted data. The security of the combined protocol (private join and private compute) follows from the security of private join, the semantic security of the threshold FHE scheme and the security of the interactive bootstrapping protocols.

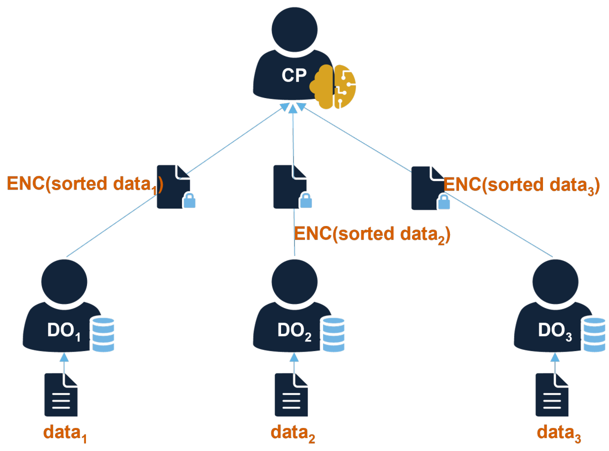


Figure 5: Diagram of the private compute. The Data Owners (DOs) encrypt (using the joint threshold FHE public key) and send their sorted data records (the sort corresponds to the index list received from the CP in the private join protocol) to the Computation Party (CP). CP extracts the encrypted intersection of the records and performs computations over it. The last step of the private computation (not depicted) involves the CP decrypting the obtained encrypted result with the help of the DOs, who hold the secret key shares.

Table A1: Kaplan-Meier analysis results for the first 15 weeks following initial oxaliplatin treatment for the data set of colorectal cancer patients; SD denotes survival data.

Time	Patients Treated with Cannabis		Patients Treated without Cannabis	
(weeks)	SD in the clear	encrypted SD	SD in the clear	encrypted SD
0	0.997	0.997	1.000	1.000
1	0.985	0.985	0.990	0.99
2	0.967	0.967	0.969	0.969
3	0.952	0.952	0.962	0.962
4	0.937	0.937	0.944	0.944
5	0.922	0.922	0.916	0.916
6	0.919	0.919	0.913	0.913
7	0.907	0.907	0.885	0.885
8	0.889	0.889	0.878	0.878
9	0.88	0.880	0.868	0.868
10	0.871	0.871	0.840	0.840
11	0.862	0.862	0.826	0.826
12	0.853	0.853	0.812	0.812
13	0.841	0.841	0.801	0.801
14	0.823	0.823	0.784	0.784
15	0.802	0.802	0.774	0.774

Note that this Kaplan-Meier analysis has no clinical significance and should not be interpreted as such. The analysis was performed solely for the purpose of testing the proposed FHE method.

G Multiparty Setup

Although all of the computations are based on the same public data set, several different filters are applied depending on the data owner/computation. To simplify the process, we provide the tables that would be produced by each data owner after filtering in the clear (without any knowledge of the second data set). The actual data on which the statistics are performed come from the matches obtained using the *private join and compute* protocol. The computation is performed using multiparty FHE ensuring the confidentiality of the information provided by each data owner.

The use case we address is when one owner has the patient/genetic data and the other owner has the outcomes data; both parties are interested in analyzing the relative effectiveness of the treatment. To simulate it, the reference data set `data /S1_Clinical_and_Immune_Data.csv` is used to generate two data sets that represent the patient/genetic data, and the second one the treatment outcome, called *Data Owner 1* (DO_1) and called *Data Owner 2* (DO_2), respectively.

The splitting of the data between the data owners and the fields used for each computation is described in Table A2.

Table A2: Splitting of the public data set [9] between two data owners; only one data owner was involved in FHE computations for mean, median, standard deviation, frequency, and t-test1, which corresponds to the single-key FHE scenario.

	Data Owner 1					Data Owner 2							
Computation	SUBJID	age	sex	arm	PBRM1	SUBJID	ORR	OS	OS_CNSR	PS	PS_CNSR	# prior therapies	benefit
mean		✓											
median		✓											
standard deviation		✓											
frequency 1			✓										
frequency 2													✓
frequency 3					✓								
frequency 4												✓	
χ^2 test				✓			✓						
t-test 1		✓		✓									
t-test 2		✓					✓						
KM & LR 1		✓						✓	✓				
KM & LR 2			✓					✓	✓				
KM & LR 3					✓					✓	✓		
logistic regression		✓	✓	✓			✓						

SUBJID: subject Id; KM: Kaplan-Meier survival, LR: log-rank survival; ORR: objective response rate; OS: overall survival time; OS_CNSR: overall survival censor; PFS: progression-free survival time; PFS_CNSR: progression-free survival censor.

Data sets

DO_1 filters out all the rows in which any of the following attributes are not available: `SUBJID, Age`, `Sex, Arm`. DO_2 keeps only complete rows (where there are no missing values). The results correspond to the data sets `DO_1.csv` and `DO_2.csv`. The number of rows for each of them are `985` and `979`, respectively. The size of the intersection is `958`. The column used to identify each patient is `SUBJID`.

To emulate the real scenario, the rows of each data set were randomly permuted. Column normalization was also performed and added to the corresponding table.

Additional two tables are `DO_1B.csv` and `DO_2C.csv`. The former corresponds to the cases in which `Arm` is equal to `NIVOLUMAB` and `PBRM1` equal to 1. The latter is obtained by retaining the rows of DO_2 where `ORR` is available. The numbers of rows in each of them are `255` and `898`, respectively.

For conciseness, the following tables are introduced

- **table_A:** inner-join of `DO_1.csv` and `DO_2.csv` of size `958`
- **table_B:** inner-join of `DO_1B.csv` and `DO_2.csv` of size `230`
- **table_C:** inner-join of `DO_1.csv` and `DO_2C.csv` of size `883`

In the multiparty FHE computation, the inner-join is replaced by the *private join and compute* protocol.

Computations

Table A3 shows the tables, the field(s) computed on, and the plaintext results for all computation scenarios.

Computation	`table_A`	`table_B`	`table_C`	Result
Mean	age			61.279
Median	age			62
Standard deviation	age			10.179
Frequency 1	sex			243, 715
Frequency 2	benefit			278, 383, 297
Frequency 3	PBRM1			130, 324, 552
Frequency 4	# prior therapies			10, 559, 297, 90, 2
χ^2			ORR & arm	1.062e-13
t-test 1	age & arm			0.048
t-test 2	age & ORR			0.371
KM & LR 1	OS, `OS_CNSR`, arm			0.001
KM & LR 2	OS, `OS_CNSR`, sex			0.104
KM & LR 3		PFS,`PFS_CNSR`, PBRM1		0.006
Log Reg			target: ORR	7.287e-09
			age, sex, arm	0.145
				0.074
				1.186e-11

Table A3: Summary of joined tables, fields the computations were performed on, and plaintext results.

	FHE	Reference	Abs Error	Rel Error
mean	61.2787056369	61.2787056367	1.74e-10	2.83e-12
median	62.0000000000	62.0000000000	0.00e+00	0.00e+00
standard deviation	10.1793009606	10.1792993071	1.65e-06	1.62e-07

Table A4: Relative and absolute error of mean, median and standard deviation

H Accuracy of Descriptive Statistical Computations

In this section we compare the accuracty of FHE results vs results in the clear.

Since the number of multiplication levels required by the statistical computations discussed in this section is relatively low, we use leveled FHE (without bootstrapping). All of the computations, except the quantile (for the median) and Kaplan-Meier survival analysis, are computed as described by the standard textbooks. Most of the computations are implemented using CKKS; the only exceptions are the `frequencies`, which are implemented with BFV.

Number of prior therapies	FHE	Reference	error
0	10	10	0
1	559	559	0
2	297	297	0
3	90	90	0
4	2	2	0

Table A5: Frequency analysis for the number of prior therapies

Sex	FHE	Reference	error
F	243	243	0
M	715	715	0

Table A6: Number of rows per sex

Benefit	FHE	Reference	error
CB	278	278	0
ICB	383	383	0
NCB	297	297	0

Table A7: Number of rows per clinical benefit type

PBRM1	FHE	Reference	error
MUT	116	116	0
WT	304	304	0
	538	538	0

Table A8: Number of rows per somatic mutation

	FHE	Reference	Abs Error	Rel Error
statistic	5.524964962e+01	5.524964951e+01	1.05e-07	1.91e-09
p-value	1.061532267e-13	1.061532324e-13	5.70e-21	5.37e-08

Table A9: Pearson's χ^2 test

	FHE	Reference	Abs Error	Rel Error
average_0	62.05655528	62.0565552700	1.26e-08	2.03e-10
average_1	60.75503357	60.7550335600	1.00e-08	1.65e-10
p-value	0.047107877	0.0471078780	3.46e-10	7.35e-09
t-score	1.988205929	1.9882059260	3.13e-09	1.57e-09

Table A10: *t*-test of Age by trial arm

	FHE	Reference	Abs Error	Rel Error
average_0	61.05570653	61.0557065200	1.04e-08	1.71e-10
average_1	61.87074831	61.8707483000	1.21e-08	1.95e-10
p-value	0.370745113	0.3707451140	7.63e-10	2.06e-09
t-score	-0.896996543	-0.8969965410	1.44e-09	-1.60e-09

Table A11: *t*-test of Age by ORR groups

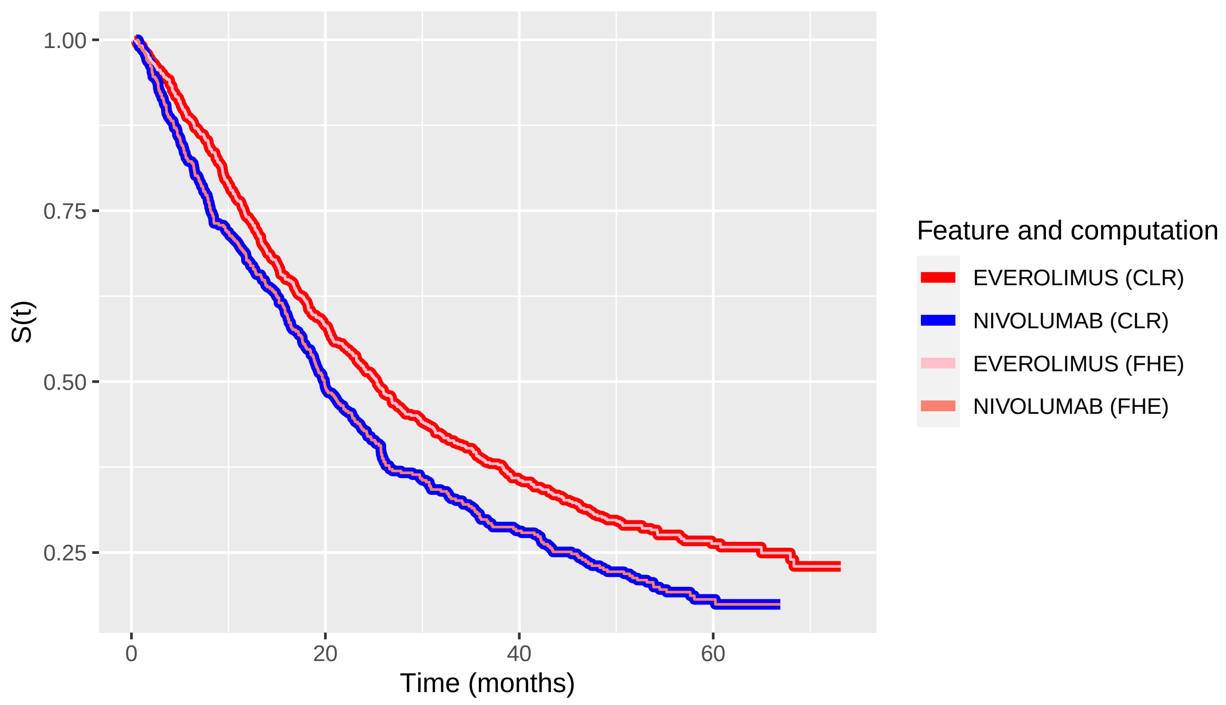


Figure 6: Survival Associated with Treatment (NIVOLUMAB: arm = 1 vs EVEROLIMUS: arm = 0). Note that there are four curves displayed, the thick ones correspond to computation in the clear and the thin ones to the FHE case.

I Accuracy of Survival Analysis computations

Kaplan-Meier and log-rank analyses are performed for *Arm* and *Sex* for *overall survivability* and PBRM1 for *progression-free survival*. The corresponding time & censor columns are (OS, OS_CNRS) and (PFS, PFS_CNRS), respectively.

Survival 1: `OS` associated with `Arm` treatment

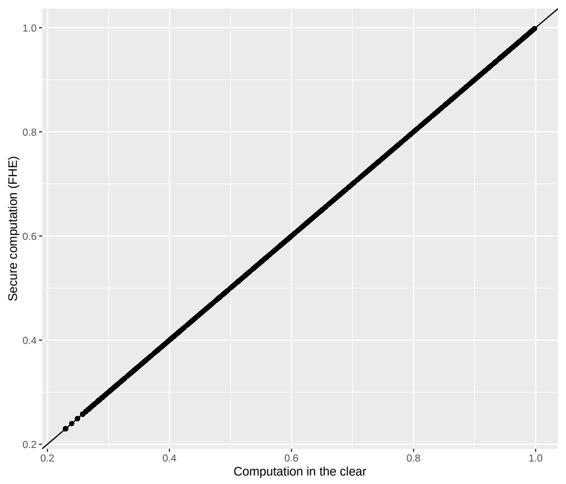
This section presents the Kaplan-Meier plot of survival probability per `arm` over time `OS`. The corresponding log-rank test is subsequently presented.

Figure 6 presents two pairs of Kaplan-Meier plots. This graph depicts the superposition of the results in the clear with lines in `red` and `blue` along with the computed using FHE, represented with thick lines in `pink` and `salmon`.

An alternative way to examine the accuracy is to plot the FHE result as a function of the expected result (the computation in the clear), which is illustrated in figure 7.

	FHE	Reference	Abs Error	Rel Error
χ^2	10.5979814473	10.5979815589	1.12e-07	1.05e-08
p-value	0.0011321119	0.0011321118	6.84e-11	5.37e-08

Table A12: log-rank test of survival probability per `arm` over time `OS`



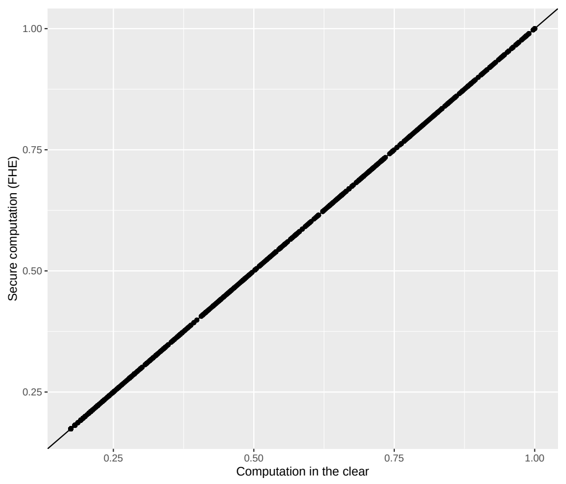


Figure 7: FHE-computed survival Survival Associated with `NIVOLUMAB` (on the left) and `EVEROLIMUS` (on the right) treatment as a function of the expected result. The maximal differences between the two curves are 2.19×10^{-7} and 3.76×10^{-7}, respectively

Survival 2: `OS` associated with `Sex`

This section presents the Kaplan-Meier plot of survival probability per `Sex` over time `OS`. The corresponding log-rank test is subsequently presented.

	FHE	Reference	Abs Error	Rel Error
χ^2	2.6357065143	2.6357063963	1.18e-07	4.48e-08
p-value	0.1044855720	0.1044855798	7.43e-08	5.37e-08

Table A13: log-rank test of survival probability per `Sex` over time `OS`

Figure 8 presents two pairs of Kaplan-Meier plots. This graph depicts the superposition of the results in the clear with lines in `red` and `blue` along with the computed using FHE represented with thick lines in `pink` and `salmon`.

An alternative way to examine the accuracy is to plot the FHE result as a function of the expected result (the computation in the clear), which is illustrated in figure 9.

Survival 3: `PFS` associated with `PBRM1` somatic mutation

This section presents the Kaplan-Meier plot of survival probability per somatic mutation `pbrm1` over time `PFS`. The corresponding log-rank test is subsequently presented.

Figure 10 presents two pairs of Kaplan-Meier plots. This graph depicts the superposition of the results in the clear with lines in `red` and `blue` along with the computed using FHE represented with thick lines in `pink` and `salmon`.

An alternative way to examine the accuracy is to plot the FHE result as a function of the expected result (the computation in the clear), which is illustrated in figure 11.

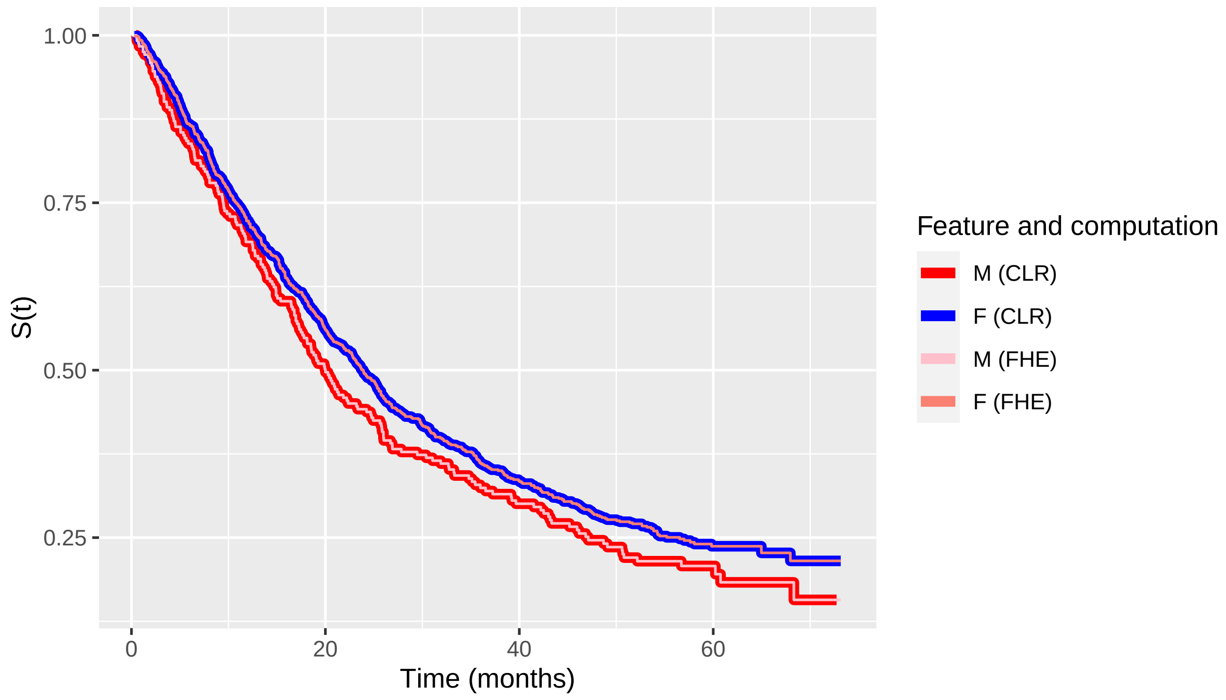


Figure 8: Survival Associated with Sex. Note that there are four curves displayed; the thick ones correspond to computation in the clear and the thin ones to the FHE case.

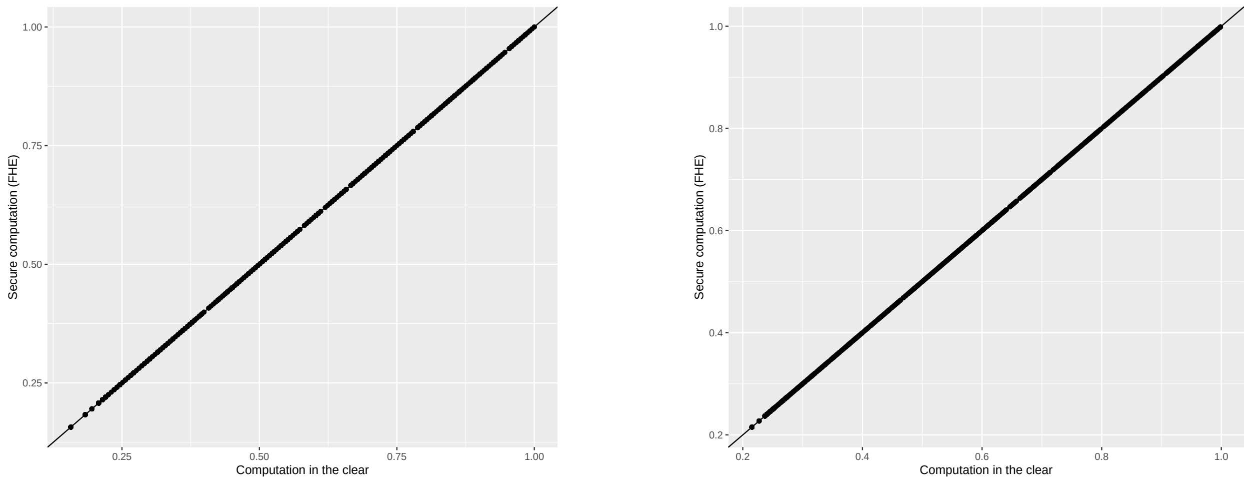


Figure 9: FHE-computed Survival Associated with `F` (on the left) and `M` (on the right) as a function of the expected result. The maximal differences between the two curves are 8.39×10^{-8} and 4.76×10^{-7}, respectively

	FHE	Reference	Abs Error	Rel Error
χ^2	7.6306513414	7.6306512305	1.11e-07	1.45e-08
p-value	0.0057384575	0.0057384579	3.53e-10	6.15e-08

Table A14: log-rank test of survival probability per `pbrm1` over time `PFS`

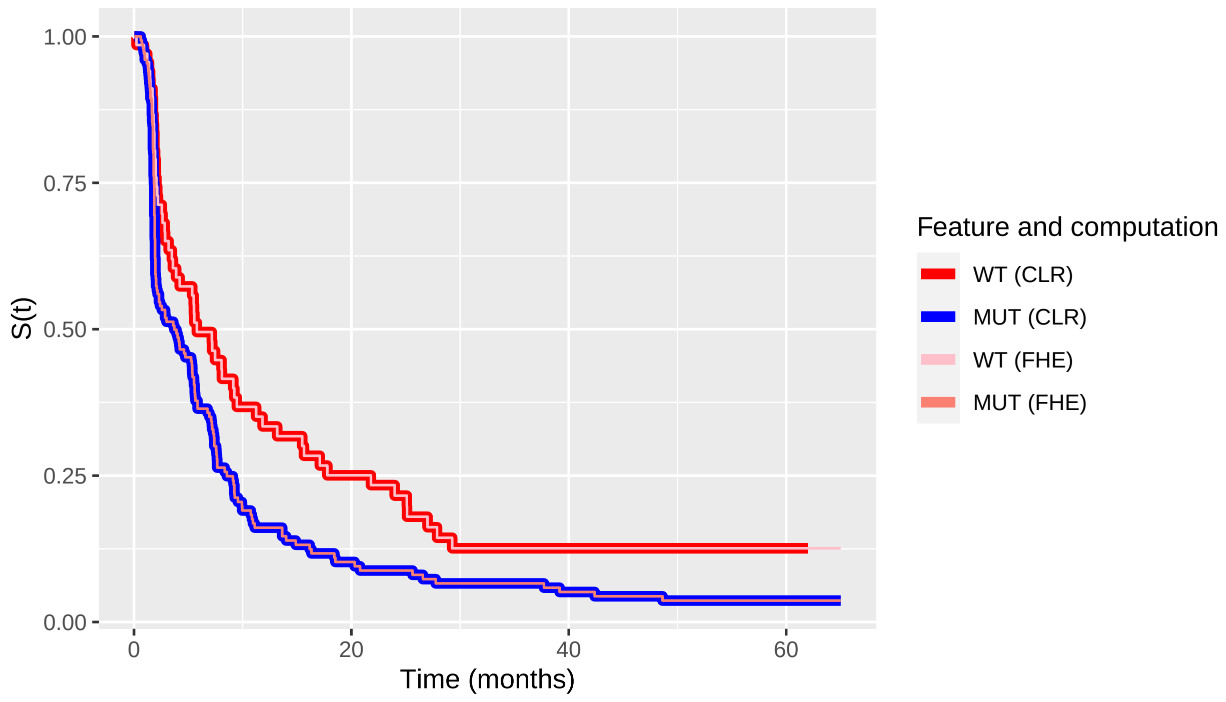
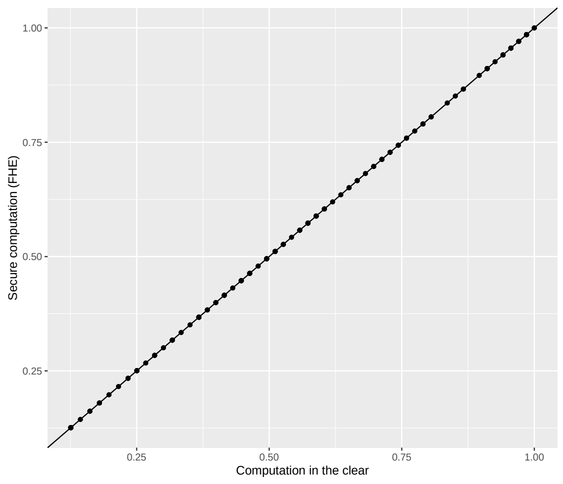


Figure 10: Survival with Somatic Mutation after immunotherapy. Note that there are four curves displayed, the thick ones correspond to computation in the clear and the thin ones to the FHE case.



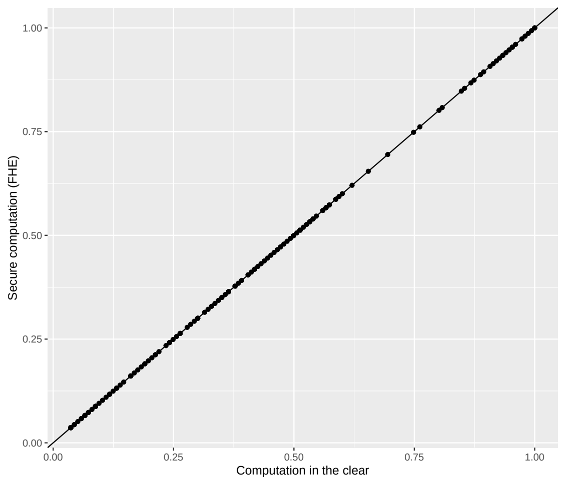


Figure 11: FHE-computed Survival Associated with `MUT` (on the left) and `WT` (on the right) treatment as a function of the expected result. The maximal differences between the two curves are 1.73×10^{-8} and 2.68×10^{-7}, respectively

Iters	Intercept	Abs Error	Rel Error	pvalue	Abs Error	Rel Error
10	-3.5605359970	0.1573837130	4.23e-02	1.74e-08	1.01e-08	1.38e+00
20	-3.6994260920	0.0184936180	4.97e-03	8.06e-09	7.70e-10	1.06e-01
30	-3.7155884840	0.0023312260	6.27e-04	7.38e-09	9.27e-11	1.27e-02
40	-3.7176235140	0.0002961960	7.97e-05	7.30e-09	1.17e-11	1.61e-03
50	-3.7178820390	0.0000377000	1.01e-05	7.29e-09	1.48e-12	2.04e-04
60	-3.7179149290	0.0000047800	1.29e-06	7.29e-09	1.89e-13	2.59e-05
70	-3.7179191140	0.0000005970	1.60e-07	7.29e-09	1.89e-14	2.60e-06
80	-3.7179196480	0.0000000621	1.67e-08	7.29e-09	2.64e-16	3.62e-08
90	-3.7179197250	0.0000000146	3.92e-09	7.29e-09	4.27e-15	5.86e-07
100	-3.7179197330	0.0000000229	6.16e-09	7.29e-09	1.46e-15	2.00e-07

Table A15: Evolution of `Intercept` with the `number of iterations`

J Logistic Regression Training

The parameters for the logistic regression are obtained by the maximization of the likelihood function. As there is no analytic solution for it, a numerical approach is used in practice. We used a fixed-Hessian Newton-Raphson method, which avoids matrix inversion (performs it only once at the beginning) and does not require a tunable parameter. The fixed-Hessian method was proposed in [11], where it was referred to as the lower-bound method, based on convergence analysis of the Newton-Raphson algorithm.

We used the following approximation and FHE parameters:

- **Sigmoid function approximation**: Chebyshev polynomial of degree 64, for the range [-4,4].
- **Multiplication depth**: 14 levels per iteration.
  - Polynomial evaluation: 8 levels.
  - Matrix multiplication: 4 levels corresponding to two matrix multiplications
  - Interactive bootstrapping: 1 level (performed at the end of each iteration).
  - 1 level for final rescaling of the coefficients.

The FHE scheme was CKKS with at least 128 bits of security (ring dimension $N = 2^{16}$).

Numerical Accuracy

The tables below illustrate the performance measurements (numerical accuracy and computational) after every 10 iterations (up to 100 iterations). For each measurement, the coefficients, p-values, and their errors with respect to `statsmodels` results are provided.

Iters	Age	Abs Error	Rel Error	pvalue	Abs Error	Rel Error
10	0.0132689580	0.0004115500	3.01e-02	1.53e-01	8.56e-03	5.92e-02
20	0.0136372780	0.0000432000	3.16e-03	1.45e-01	8.40e-04	5.81e-03
30	0.0136751220	0.0000053900	3.94e-04	1.45e-01	1.04e-04	7.19e-04
40	0.0136798240	0.0000006840	5.00e-05	1.45e-01	1.32e-05	9.13e-05
50	0.0136804220	0.0000000859	6.28e-06	1.45e-01	1.64e-06	1.13e-05
60	0.0136804970	0.0000000108	7.89e-07	1.45e-01	2.06e-07	1.43e-06
70	0.0136805080	0.0000000002	1.29e-08	1.45e-01	2.05e-08	1.42e-07
80	0.0136805090	0.0000000007	5.15e-08	1.45e-01	2.49e-08	1.72e-07
90	0.0136805090	0.0000000013	9.63e-08	1.45e-01	4.33e-08	3.00e-07
100	0.0136805080	0.0000000001	9.89e-09	1.45e-01	8.70e-09	6.02e-08

Table A16: Evolution of `Age` with the `number of iterations`

Iters	Genter	Abs Error	Rel Error	pvalue	Abs Error	Rel Error
10	-0.4061559100	0.0158741150	3.76e-02	8.13e-02	7.74e-03	1.05e-01
20	-0.4203994540	0.0016305710	3.86e-03	7.43e-02	7.25e-04	9.85e-03
30	-0.4218277280	0.0002022970	4.79e-04	7.37e-02	8.91e-05	1.21e-03
40	-0.4220043630	0.0000257000	6.08e-05	7.36e-02	1.13e-05	1.53e-04
50	-0.4220267570	0.0000032700	7.74e-06	7.36e-02	1.43e-06	1.95e-05
60	-0.4220296040	0.0000004210	9.97e-07	7.36e-02	1.87e-07	2.54e-06
70	-0.4220299680	0.0000000568	1.35e-07	7.36e-02	2.00e-08	2.72e-07
80	-0.4220300150	0.0000000099	2.34e-08	7.36e-02	2.58e-09	3.50e-08
90	-0.4220300230	0.0000000025	6.03e-09	7.36e-02	2.74e-09	3.72e-08
100	-0.4220300160	0.0000000089	2.10e-08	7.36e-02	9.45e-09	1.28e-07

Table A17: Evolution of `gender` with the `number of iterations`

Iters	Arm	Abs Error	Rel Error	pvalue	Abs Error	Rel Error
10	-0.4061559100	0.0158741150	3.76e-02	8.13e-02	7.74e-03	1.05e-01
20	-0.4203994540	0.0016305710	3.86e-03	7.43e-02	7.25e-04	9.85e-03
30	-0.4218277280	0.0002022970	4.79e-04	7.37e-02	8.91e-05	1.21e-03
40	-0.4220043630	0.0000257000	6.08e-05	7.36e-02	1.13e-05	1.53e-04
50	-0.4220267570	0.0000032700	7.74e-06	7.36e-02	1.43e-06	1.95e-05
60	-0.4220296040	0.0000004210	9.97e-07	7.36e-02	1.87e-07	2.54e-06
70	-0.4220299680	0.0000000568	1.35e-07	7.36e-02	2.00e-08	2.72e-07
80	-0.4220300150	0.0000000099	2.34e-08	7.36e-02	2.58e-09	3.50e-08
90	-0.4220300230	0.0000000025	6.03e-09	7.36e-02	2.74e-09	3.72e-08
100	-0.4220300160	0.0000000089	2.10e-08	7.36e-02	9.45e-09	1.28e-07

Table A18: Evolution of `Arm` with the `number of iterations`

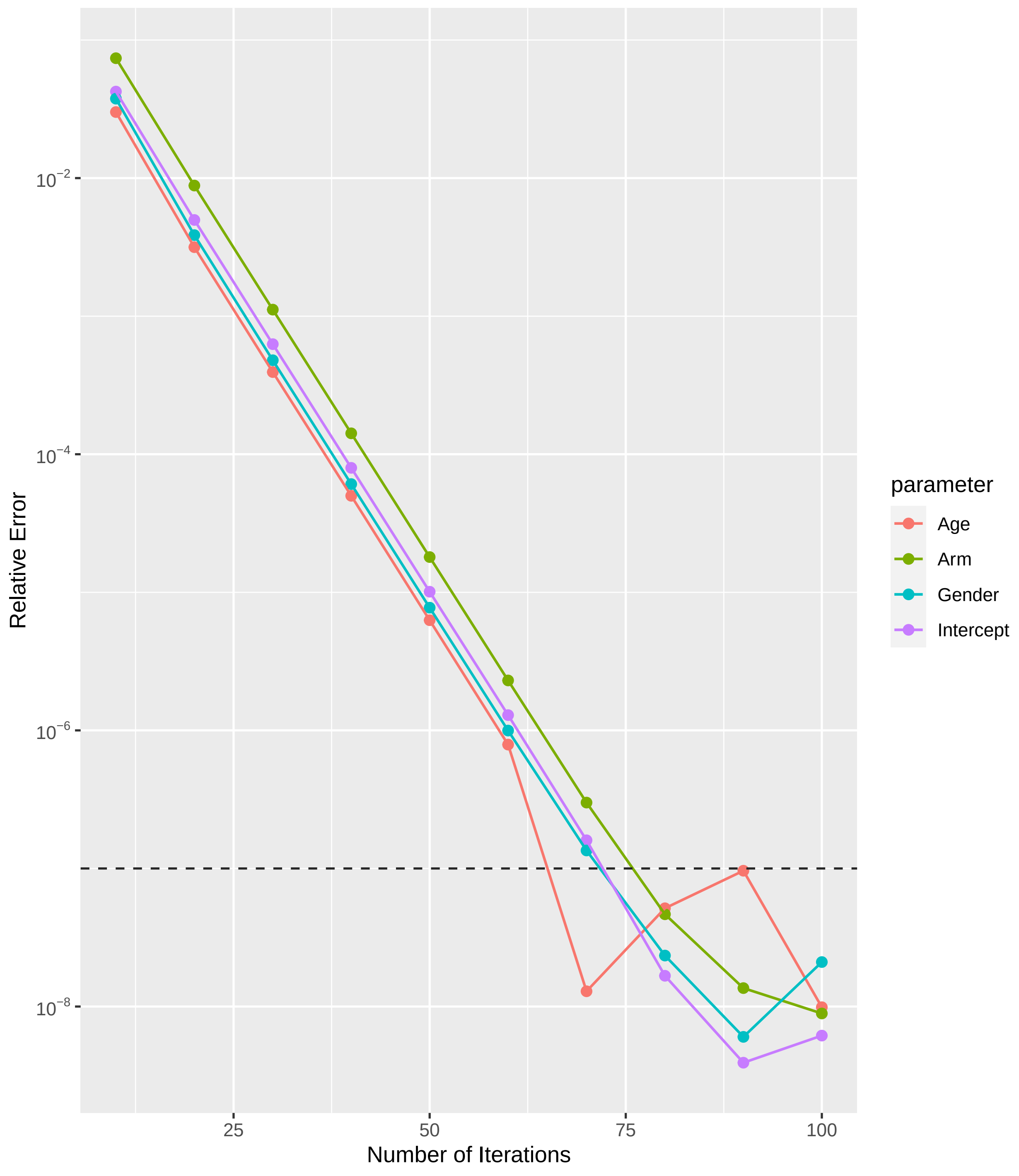


Figure 12: Evolution of the relative error of the `computed parameters` with the `number of iterations`. The dashed line corresponds to the numerical accuracy estimated in the previous section.

Approach	AUC	-log-likelihood	sigmoid range
GLM	0.95978	0.2236	37
GLMNET	0.95976	0.2237	111

Table A19: Performance results for standard R packages

Effect of polynomial degree and range in sigmoid approximation on accuracy of logistic regression training

We illustrate that relatively high polynomial degrees and ranges in approximating the sigmoid function using the Chebyshev interpolation are needed to achieve satisfactory accuracy results for logistic regression training. We use the CDC 2014 infant mortality data to perform this analysis. We chose this data set instead of the oncological [9], in order to work with larger numbers of samples and features, i.e., the scenarios in which logistic regression is typically used.

To prepare for the analysis, we performed the following pre-processing steps. All positive cases were kept and an equal number of randomly selected negative cases were selected, giving a data set of 46,594 points. From this, 80% were used for training (37,265 rows) and the remaining 20% (9,319 rows) were used for testing. The target variable is IMORT and the independent variables are: BWTR14, RECWT, APGAR5R, BFEDN, AB_NICUN, APGAR5, CIG_RECN, CA_CCHDY and BFEDU.

As a reference, we trained a logistic regression on this data using two standard R packages: GLM and GLMNET (Table A19). We recorded the Area Under Curve (AUC), negative (-) log-likelihood, and maximum range of the input to the sigmoid function for both cases (range for the sigmoid function corresponds to [-range/2,range/2]).

Note that although the AUC and the negative log-likelihood are the same from the statistical point of view, the ranges (the maximal argument that the sigmoid function may get) are significantly different between the packages.

Next, we examined three different methods for logistic regression training for the actual sigmoid function (without any approximation): Newton-Raphson, fixed-Hessian (what we used for the experiments in this paper), and Nesterov Accelerated Gradient (NAG) descent (used in [31]). The results are illustrated in Table A20. We see that while the number of iterations can vary significantly, the AUC and negative log-likelihood are reasonably close to the results achieved with standard packages. The parameters for the NAG method were tuned to minimize the number of iterations for this scenario. For the stopping criteria we used the variation of the average log-likelihood between two successive iterations becoming less than 10^{-4}.

To determine the optimal parameters when using the fixed-Hessian approach, we generated a grid of parameters as displayed in Figure 13. We observe that out of the 144 possible combinations, the fixed-Hessian method converged in 79 cases, diverged in 52 and failed to reach the convergence criterion in 13 cases. The lowest polynomial degree at which the benchmark AUC was achieved was 32 and range was 64. The number of necessary iterations to reach the convergence criteria was 117.

The same setup was used to find the optimal parameters for the NAG method. The results are illustrated in Figure 14. We observe that out of 144 possible combinations, the NAG iterations converged in 74 cases, diverged in 52 and did not reach the convergence criterion in 18 cases. The lowest polynomial degree for which the benchmark AUC was achieved was 32 and range was 128.

Approach	Number of iterations	AUC	-log-likelihood	sigmoid range
Newton	14	0.95984	0.2234	560
Fixed-Hessian	308	0.95969	0.2328	73
NAG	124	0.95913	0.2316	125

Table A20: Performance results for three common logistic regression training methods.

The number of necessary iterations was 75 (using the same convergence criteria as the previous case). The NAG momentum and learning rate parameters were optimized to minimize the number of iterations.

The above analysis implies that a degree-3 polynomial approximation, such as the one used in [31], is not expected to satisfy accuracy requirements for many practical scenarios as polynomial degrees for other data sets may be much larger, e.g., 32 for the infant mortality data set.

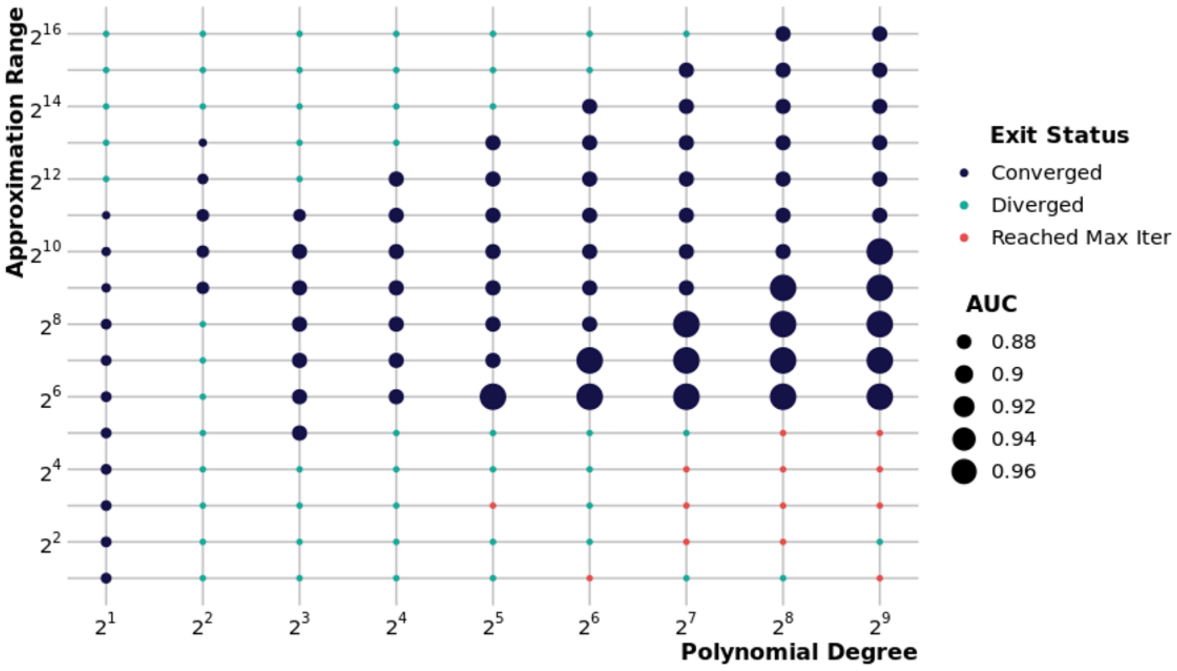


Figure 13: Results for the fixed-Hessian method with the sigmoid function approximated using a Chebyshev interpolation. The x axis indicates the sigmoid approximation polynomial degree 2 to 512 and the y axis indicates the input range of 2 to 65,536, using powers of two in both cases. The graph uses a color code to indicate the exit status (converged, diverged, or terminated by reaching the maximal number of iterations), and the size of the point is related to the AUC of the resulting model (bigger is better)

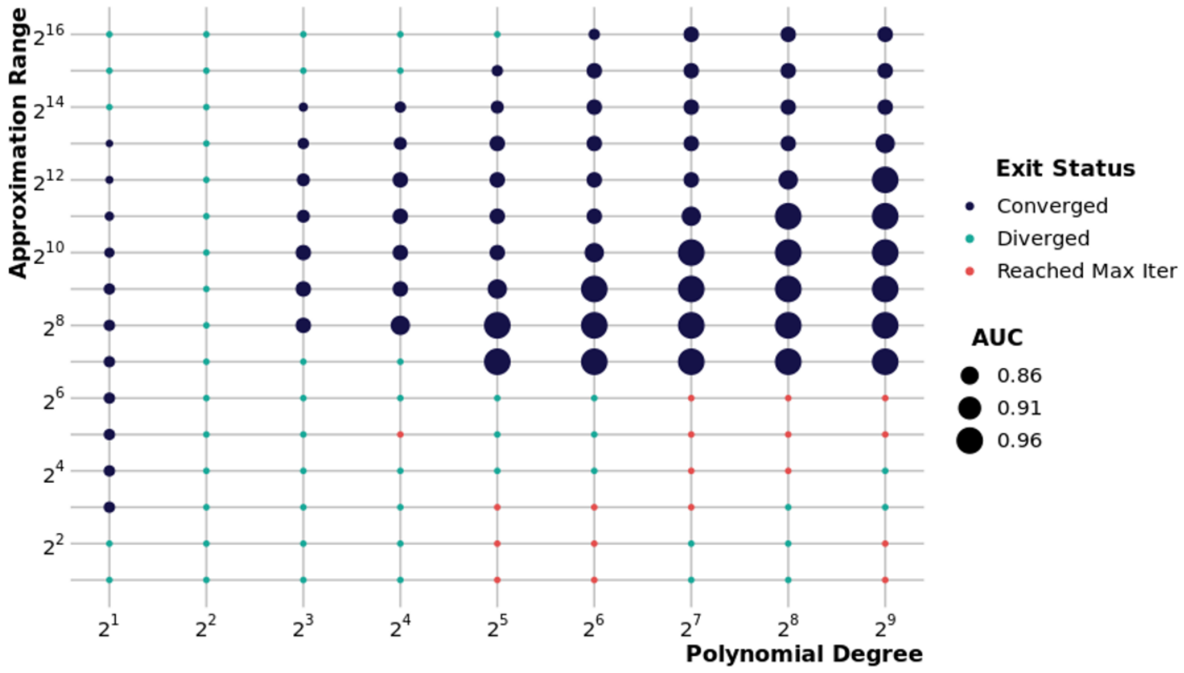


Figure 14: Results for the NAG method with the sigmoid function approximated using a Chebyshev interpolation. The x axis indicates the sigmoid approximation polynomial degree 2 to 512 and the y axis indicates the input range of 2 to 65,536, using powers of two in both cases. The graph uses a color code to indicate the exit status (converged, diverged, or terminated by reaching the maximal number of iterations), and the size of the point is related to the AUC of the resulting model (bigger is better)

K Glossary of Terms

Term	Definition
Biomarker	A biological indicator of a normal or abnormal process, or of a condition or disease.
Chebyshev Interpolation	A numerical method of approximating a (possibly non-polynomial) function using a polynomial.
CheckMate x	Clinical trials that compared the efficacy and safety of different cancer treatments in patients with advanced renal cell carcinoma (RCC).
Chi-Squared Test	A statistical test that can be used to determine whether there is a significant difference between the expected and observed frequencies in one or more categories of a contingency table.
Data Anonymization	The process of removing or altering personal identifying information from data so that individuals cannot be identified.
Exhausted Ciphertext	A ciphertext that has been subjected to so many homomorphic operations that it can no longer be used for further computation without refreshing.
Federated Collaboration Model	A learning technique that trains a model on multiple data sets that remain on the devices where they were collected or generated.
Fully Homomorphic Encryption	An encryption scheme that allows arbitrary computations to be performed on encrypted data without decryption.
Garbled circuit	A computation circuit with randomly masked inputs that can only be evaluated by an interactive protocol among the participants without a trusted party.
Interactive Bootstrapping	A technique used in multiparty fully homomorphic encryption schemes to refresh the ciphertexts to allow further computation.
Kaplan-Meier Survival Analysis	A non-parametric statistic used to measure the fraction of patients living for a certain amount of time after a certain treatment.
Learning Rate	A hyperparameter that is used in machine learning to control the size of the steps taken by an optimization algorithm during the training phase.
Logistic Regression	A statistical model that is used to predict the probability of a binary response variable, such as a variable taking the values yes or no, given a set of predictor variables.
Log-Rank Test	A statistical test that is used to compare the survival distributions of two or more groups.
mTor inhibitors	A class of drugs that target the mammalian target of rapamycin (mTOR), a protein that plays a role in cell growth and division.
MultiParty Computation	A cryptographic technique that allows multiple parties to jointly compute a function on their private data without revealing their individual data to each other nor to a trusted party.
Nesterov Gradient Descent	An optimization algorithm that uses momentum to accelerate convergence to optimal points and prevent getting trapped in local minima.
Nivolumab	A drug that attaches to the protein PD-1 to help the immune system fight cancer cells more effectively and it is used to treat many different types of cancer.
Objective Response Rate	A measure of how well a cancer treatment works.
Oxaliplatin	A chemotherapy-based medical drug.
Programmed Death 1 (PD-1)	A protein that is present on T immune cells that helps to regulate the body's immune responses.
Randomized Controlled Trial	Type of scientific experiment in which participants are randomly assigned to a number of groups. One group receives the treatment being tested, while the other groups receive either a placebo (fake treatment) or an alternative treatment.
Renal Cell Carcinoma	A common type of kidney cancer in adults.
Secret Sharing	A cryptographic technique that allows a secret to be divided into multiple portions (shares), such that only a designated number of shares are needed to reconstruct the secret.
Single Nucleotide Polymorphisms	Variations in DNA that occur at specific locations in the genome.
Student's T Test	A statistical test that can be used to compare the means of two groups or populations.
Survival Analysis	A statistical method for analyzing the expected time to an event. The event can be death or disease progression.
Threshold Fully Homomorphic Encryption	A fully homomorphic encryption scheme that allows multiple parties to contribute encrypted private data for homomorphic computation, with the ability of decrypting the ciphertext results collectively by a designated group of users.

Table A21: Glossary of terms.