PEA: Practical Private Epistasis Analysis using MPC

Kay Hamacher[1] [0000−0002−6921−8345], Tobias Kussel[*,1] [0000−0002−9731−7666], Thomas Schneider[1] [0000−0001−8090−1316], and Oleksandr Tkachenko[*,(✉),2] [0000−0001−9232−6902]

[1] Technical University of Darmstadt, Germany
[2] DFINITY

Abstract. Due to the significant drop in prices for genome sequencing in the last decade, genome databases were constantly growing. This enabled genome analyses such as Genome-Wide Association Studies (GWAS) that study associations between a gene and a disease and allow to improve medical treatment. However, GWAS fails at the analysis of complex diseases caused by non-linear gene-gene interactions such as sporadic breast cancer or type 2 diabetes. Epistasis Analysis (EA) is a more powerful approach that complements GWAS and considers non-linear interactions between multiple parts of the genome and environment.
Statistical genome analyses require large, well-curated genomic datasets, which are difficult to obtain. Hence, the aggregation of multiple databases is often necessary, but the sharing of genomic data raises severe privacy concerns and is subject to extensive regulations (e.g., GDPR or HIPAA), requiring further privacy protection for collaborative analyses.
Although there has been work on private GWAS, there was a lack of attention to Private EA (PEA). In this work, we design the first secure and accurate PEA protocol, with security against passive adversaries.
Our efficient PEA protocol consists of two subprotocols: (1) (optional) feature selection for filtering noisy features to reduce the input size for better efficiency and (2) finding relevant associations. For feature selection, we design two protocols based on Secure Multi-Party Computation (MPC) for Relief-F and TuRF. For finding associations, we design an MPC protocol for Multifactor Dimensionality Reduction (MDR).
Our private MDR protocol is based on two novel, efficient building blocks, *arithmetic greater than* and *arithmetic swap*, which may be of independent interest. This approach omits the need for expensive conversions between sharing types in private MDR and reduces the communication by two orders of magnitude compared to a naïve design using garbled circuits. Our private MDR protocol runs in (extrapolated) three days on a practical database with 10,000 features for all two mutually combined features, i.e., considering about 50 million combinations.

Keywords: epistasis · genomic privacy · MPC

(✉) Corresponding author: Oleksandr Tkachenko (oleksandr.tkachenko1@gmail.com); the work presented in this paper was mainly done while the author was a doctoral researcher at Technical University of Darmstadt
[*]Equally contributed

1 Introduction

Technical advances and reduced costs in genome sequencing technology will allow full genome sequencing to become a standard medical procedure in the near future. This plethora of genomic data opens up interesting possibilities not only in personalized treatment of diseases, but in a research context as well. Genome Wide Association Studies (GWAS) allow to statistically link a small number of genetic variants (e.g., Single Nucleotide Poymorphisms (SNPs)) to a phenotypical trait, which in medical research is often the manifestation of a disease like diabetes, hypertension, or cancer.

Due to the sensitive nature of genomic data, which is the ultimate personal identifier [18], and the vast amount of required data, the need for privacy preserving analysis methods arises. Non-genomic medical patient data is often analyzed in anonymized or pseudonymized form. Unfortunately, these traditional and other approaches like statistical disclosure control [13] are difficult to apply correctly and in most cases unsuitable for genomic data. More evolved statistical disclosure methods, like differential privacy, suffer from some loss of utility when applied to genomic data due to the inherent interactivity in, e.g., tumor boards.

Secure Multi-Party Computation (MPC) is a class of privacy-preserving techniques that guarantees the privacy of inputs and allows the exact computation of arbitrary functionalities. MPC has successfully been applied to many different real-world problems. However, this strong privacy guarantee and flexibility comes with quite severe limitations. The required computation and communication is multiple orders of magnitude higher than in the clear text analysis. This renders this class infeasible in practice for many applications.

Following the success in applying MPC protocols to genomic analysis methods like GWAS or similar genome queries, we propose, implement, and evaluate PEA, a suite of MPC protocols that privately analyze the epistasis of SNPs in connection to the manifestation of a disease. PEA analyzes how the interaction of multiple SNPs are causally linked to the disease. This is a critical step for the development of a better understanding of a disease and new treatments. To find these higher order interactions we privately apply Multifactor-Dimensionality Reduction (MDR), Relief-F, and Tuned Relief-F Feature Selection (TuRF).

Although gene-gene and gene-environment interactions are still an actively researched area, most novel research uses established analysis methods like MDR for specific diseases [30, 46, 43] and statistical tests [29] , or adapts those methods for novel challenges, like the amount of SNPs in GWAS data sets [22]. No prior provided privacy-preserving analysis of partitioned data sets.

1.1 Related Work

Recently, a few Differential Privacy (DP) [14]-based works have been published on Private Epistasis Analysis (PEA) [7] and Private Feature Selection (PFS) [28]. However, DP relies on a trade-off between privacy and utility and cannot achieve both. Since the genomic data is exceptionally privacy-sensitive, this leads to a significant utility degradation which is a well-known problem [32].

To the best of our knowledge, we provide the first solution for PEA or PFS without utility degradation. Previous works on private Genome-Wide Association Studies (GWAS) [8, 40] serve a similar purpose as PEA but can find only correlations between single SNPs and a trait and have much smaller complexity.

1.2 Our Contributions

Our interdisciplinary work, beyond providing new research opportunities for biomedical research, makes the following contributions.

- Design and implementation of PEA, the first secure protocol that does not degrade accuracy for:
  - Relief-F [27] and TuRF [31], two popular algorithms for filtering features in Epistasis Analyses (EAs) that run in less than a day for practical database sizes containing a=10,000 SNPs and L=100 records [7].
  - MDR [36], a popular exponential-time algorithm for EA with (extrapolated) runtimes of around three days for practical database sizes containing a=10,000 SNPs [7]. The communication of our PMDR protocol is independent of the number of records.
- New efficient, generic arithmetic building blocks:
  - A 1-out-of-N Oblivious Transfer (OT) [24]-based custom protocol for Arithmetic Greater Than (AGT), i.e., a GT gate that is computed on arithmetic shares with 1.5× less communication than the state of the art [35] but 6 instead of 5 rounds of communication.
  - Arithmetic Swap (ASWAP), a generalization of the Boolean swap gates by Kolesnikov and Schneider [26] for the arithmetic case with 4× less communication than the naïve design.
  - Batched versions of both aforementioned building blocks with $O(\kappa)$ less communication, where κ is the symmetric security parameter.
- The first implementation of three-halves garbling [37][3] and its performance analysis, which shows a greater slowdown than expected due to a higher degree of branching compared to the prior best garbling scheme. Still, it optimizes for better network bandwidth which remains the bottleneck.
- The implementation of all our building blocks and protocols are integrated in the open-source repository of the MOTION framework for MPC[4].

2 Preliminaries

In this section, we describe the required basics in genomics. App. A gives an overview of Secure Multi-Party Computation (MPC) techniques used in this paper. For more details, we refer the reader to [11].

[3] https://encrypto.de/code/3H-GC
[4] https://encrypto.de/code/MOTION

2.1 Genomic Primer

The building and functioning "instructions" of all living cells are encoded in molecular form. This molecular blueprint takes in most organisms the form of the *deoxyribonucleic acid* (DNA), a double helical molecule pairing a sequence of *nucleotides*. The DNA's alphabet consists of four nucleotides (usually abbreviated by their first letter): **A**denine, **C**ytosine, **G**uanine, and **T**hymine. In double-stranded DNA (as in humans) a helix and much more involved structures (chromatin, chromosomes, etc.) are formed by physical interactions between the nucleotides of the two strands. Typically, Watson-Crick-pairing is observed, where cytosine pairs with guanine and adenine pairs with thymine forming a base pair with highest priority. This implies that the information on one strand is encoded in a 1:1 fashion on the other.

In the process of *transcription*, the defined nucleotide sequence of the DNA (viz. the *genotype*) is transcribed to a *messenger ribonucleic acid* (mRNA) molecule, which can be thought of as a working copy of a specific gene. These mRNA molecules are then *translated* to an *amino acid* sequence forming proteins necessary for the function of the cell and organism. During this translation every *codon*, that is a triplet of nucleotides, is mapped to one of the 20 standard amino acids observed in nature. Additionally there are codons coding start and stop symbols.

Some of the built proteins inhibit or promote the transcription of DNA regions encoding other proteins. These proteins are called *transcription factors* and thus are responsible for often times complex regulatory networks in which the *interaction of multiple genes* are responsible[5] for some *phenotype*, that is some observable trait (e.g., eye and hair color, or the occurrence of a disease).

The human genome consists of roughly 3.2 billion base pairs, but only 0.1 % of base pairs vary between two individuals [2]. These variants of a specific *locus*, i.e., position on the DNA strand, are called *alleles*. Due to the sparsity of the variations between two humans, it is often useful to store and use only these variations with regard to a specific reference genome. A *Singe-Nucleotide Polymorphism* (SNP) is a variation changing exactly one nucleotide, e.g., G $\rightarrow$ T. Due to the human's *diploidity*, two alleles are possible of each individual locus. For each gene (or in case of SNPs for each base) the present allele may be written in the shorthand form "AA" for the presence of the major allele on both chromosomes, "Aa" for the presence of both the major and minor allele on one of the chromosomes each and finally "aa" for the presence of the minor allele on both chromosomes.

2.2 Genome-Wide Association Studies (GWAS) and Epistasis

Genome-Wide Association Studies (GWAS) aim to link specific genotype variations to phenotype variations. More specifically often times the goal is to link SNPs to traits like the onset of specific diseases. In the first published GWAS in 2002 [34] five SNPs could be linked to various mechanisms to increase the risk of

[5] Further regulatory mechanisms, such as the influences of the chromatin structure, exist but are not of interest for this work.

myocardial infarction. In contrast to *candidate-driven* analysis, where variations of specific, pre-determined genes are analysed, GWAS analyse variations of the complete genome.

Using a labeled data set with regard to the phenotypical trait, statistical tests are performed to determine the specific SNP's likelihood to affect the trait. This likelihood is called *penetrance* and extends, as described below, to multidimensional cases. Many different statistical tests are used in practice, ranging from relatively simple *odds ratio* analysis to more involved hypothesis tests, like χ^2-*tests* [41]. In addition to the single SNP's influence, GWAS can give a starting point for finding the variation mechanism by associating the loci to known regulatory pathways [33]. As briefly described in §2.1, genes can be part of complex regulatory networks promoting or inhibiting other genes. Due to that it is unsurprising, that complex systematic diseases like cancer are not directly associated to a single gene, but are caused by possibly non-linear interactions of many genetic variables, e.g., simultaneously the presence of one SNP and the absence of two other. This interaction of genes is called *epistasis*, or *gene-gene interaction*. Unfortunately, the analysis of those interactions becomes computationally very expensive. The algorithmic complexity scales exponentially with the "interaction depth", i.e., the number of interacting genes considered.

Consequently, exact analysis methods, analysing each tuple of SNPs, are only feasible for small sections of a genome or low interaction depths. For large genome sections, or even whole genome analysis, different methods of reducing the complexity are applied. In this work we apply *(Tuned) Relief-F Feature Selection* (TuRF, cf. §2.3) and *Multifactor-Dimensionality Reduction* (MDR, cf. §2.4) to achieve privacy-preserving epistasis analysis with practical performance.

2.3 Feature selection with Relief, Relief-F and TuRF

Typical gene-gene interaction studies analyze datasets with thousands of patients but hundreds of thousands to millions of SNPs. Most of these features play no role in the expression of the phenotype of interest. The "Relief" [23] algorithm and its advancements "Relief-F" [27] (the hyphen is often omitted in literature but is present in the original work) and "Tuned ReliefF" [31] (TuRF) are feature selection algorithms to reduce the number of features by estimating the importance of a feature with respect to the training goal.

The Relief filter works by weighting the importance of a variable by comparing a randomly chosen sample (patient) with the neighboring samples. Features that are present in a neighboring sample with the same label gain weight, features present in a neighboring sample with a different label lose weight. This procedure is repeated m times. Relief-F extends this algorithm by not only sampling the nearest neighbor in both categories, but the k nearest neighbors. In the original work [27], as in our work, the number of neighbors taken into account for the weight update is $k = 10$. Furthermore, Relief-F iterates over all n entries in the dataset, instead of a randomly chosen subset of size m, i.e., $m = n$. This increases the robustness of the result against noisy features.

```
Input: Attributes $A = A_1, \dots, A_a$,
Records $R = r^1, \dots, r^n$
$W_i \leftarrow 0, \forall i$
for $i = 1 \dots m$ do
    // randomly select record
    $r^i \leftarrow\$ [r^1, r^n]$
    $H, M \leftarrow \texttt{kNN}(r^i, k)$
    for $j = 1 \dots k$ do
        for $l = 1 \dots a$ do
            $W_l \leftarrow W_l - \Delta(A_l, r^i, H_j)/m + \Delta(A_l, r^i, M_j)/m$
return $W$
```

(a) Relief-F

```
Input: Attributes $A = A_1, \dots, A_a$,
Records $R = r^1, \dots, r^n$
for $i = 1 \dots n$ do
    $W \leftarrow \texttt{ReliefF}(A, R)$
    $W \leftarrow \texttt{sort}(W)$
    // remove last $\alpha/n$ attributes
    $W \leftarrow W[0 : a - (\alpha/n)]$
    $A \leftarrow A[0 : a - (\alpha/n)]$
return $W$
```

(b) TuRF

Alg. 1: The feature selection algorithms. The original Relief algorithm only uses the nearest hit/miss, not the k nearest hits/misses. $\leftarrow\$$ indicates random sampling from a uniform distribution, m is the number or records used for feature selection, and a denotes the number of attributes. The used difference-function Δ measures *differences*. Because of that, misses are added and hits subtracted. Using TuRF, in every iteration the last (worst performing) α/n elements are removed from the respective arrays.

TuRF modifies the Relief-F filter by removing a constant fraction of the worst performing attributes after every iteration. This effectively removes the noisiest and least significant features, speeding up the computation in the subsequent calculations and increasing robustness against noisy attributes.

The formal details of the Relief-F algorithm are given in Alg. 1a. The used distance metric takes two patient records r^1, r^2 and an attribute A (in this work a locus λ) as an input and returns zero if both records have the same occurrence of the attribute, otherwise it returns one. The TuRF algorithm is given in Alg. 1b. Details of our private implementation of TuRF are given in §3.

2.4 Multifactor Dimensionality Reduction (MDR) for Epistasis Analysis

Multifactor Dimensionality Reduction (MDR) [36] is a model-free and non-parametric statistical method to detect and model epistasis. Developed in the early 2000s, it became one standard approach to model epistasis with successful identification of interactions in datasets including sporadic breast cancer, essential hypertension [30], and type 2 diabetes [9].

In short the algorithm works by categorizing a group of loci into high and low risk combinations. This effectively reduces the dimension of the interactions to one. This new one-dimensional data is then compared among each other to find the interactions that yield the lowest classification and prediction error. Usually *Leave-one-out cross validation* is used. In that cross validation approach the dataset is divided in n equally large partitions and the model is generated on $n-1$ partitions. The remaining partition is used to calculate the prediction errors. This process is repeated for all n partitions and the prediction errors are averaged to form a "final" model error. A graphical visualization of the scheme is given in Fig. 1.

3 Private Tuned Relief-F Feature Selection

As described in §2.3, the main goal of the feature selection step is to increase the weight of features (SNPs) linked to label distinction and reduce the weight of features irrelevant to these distinctions. The TuRF algorithm uses k Nearest Neighbor clustering and iterative pruning of features to achieve this goal. To avoid a biased result incurred by the ordering of the records, our TuRF implementation permutes the order of the dataset randomly. In either case only the best $a - \alpha$ features are considered in the subsequent MDR calculation. The formal description of the PReliefF protocol is given in Prot. 1. PTuRF can be seen as a straightforward extension of PReliefF, and the formal description is given in the full version of this paper. Note, that in combination with Private Multifactor Dimensionality Reduction (PMDR) (see §4), it is possible to (optionally) reveal the noisy features to reduce the input size to PMDR.

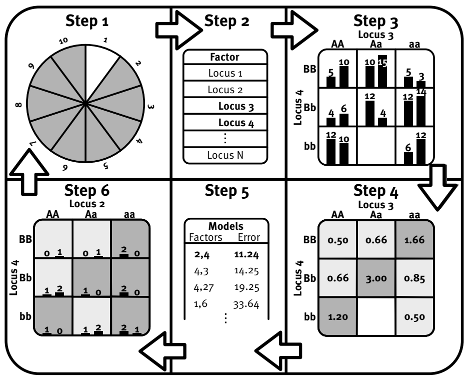

Fig. 1: High-level scheme of the MDR analysis method (adapted from [17] and [36]).

We implemented an (optional) approximation in the TuRF algorithm. Instead of recalculating the distance between the records in every iteration, the distance is considered constant, as only a small number of features are removed in every iteration. This approximation reduces the computational cost, while only incurring a small error.

Private kNN. The original Relief algorithm, as well as the improved Relief-F, require a comparison of the sampled record to the nearest neighbour or the k nearest neighbours, respectively (cf. Alg. 1a, line 6). We use adapted forms of the kNN clustering described by Järvinen et al. [20], which can be performed with a variety of metrics. Due to the comparatively low runtime cost and the nominal nature of the features, we perform a Hamming distance based clustering, i.e., based on the number of set bits (asimilar SNPs).

4 Private Multifactor Dimensionality Reduction

Private Multifactor Dimensionality Reduction (PMDR) requires aggregation of integers: data owners aggregate the counts of allele frequencies and the counts

```
1  Function PReliefF($R, \varphi$):
2      The dataset $R$ is the concatenation of each data owner's $P_i$ raw dataset $R_i$. The dataset
       consists of all records $R := (r^1, \dots, r^k)$, where the record $r^j := ((g^{j,1}, \dots, g^{j,m}), \alpha^j) : r^j \in R$
       with each genotype $g^{j,\lambda} \in \{1, 2, 3\}$ of person $j$ at locus $\lambda$ and each group $\alpha \in \{+, -\}$,
       denotes the case and control group, respectively. The function returns the index positions of
       the most weighted genotypes. $\varphi = 1 - \alpha/a$ denotes the ratio of attributes to return.
3      for $j = 1 \dots k$ do // For all records in the Dataset
           // Initialize distance and difference matrices to the numerical maximum
           // value and zero, respectively
4          $\langle m^{\text{hit}}_{\text{dist}} \rangle^Y \leftarrow [\langle \text{MAX\_VALUE} \rangle^Y, \dots, \langle \text{MAX\_VALUE} \rangle^Y]$
5          $\langle m^{\text{hit}}_{\text{ineq}} \rangle^Y \leftarrow [[\langle 0 \rangle^Y, \dots, \langle 0 \rangle^Y], \dots, [\langle 0 \rangle^Y, \dots, \langle 0 \rangle^Y]]$
6          $\langle m^{\text{miss}}_{\text{dist}} \rangle^Y \leftarrow [\langle \text{MAX\_VALUE} \rangle^Y, \dots, \langle \text{MAX\_VALUE} \rangle^Y]$
7          $\langle m^{\text{miss}}_{\text{ineq}} \rangle^Y \leftarrow [[\langle 0 \rangle^Y, \dots, \langle 0 \rangle^Y], \dots, [\langle 0 \rangle^Y, \dots, \langle 0 \rangle^Y]]$
8          for $i > j$ do // For all pairs of records
9              $\langle D_{ji} \rangle^Y \leftarrow \emptyset$
10             for $\lambda = 1 \dots m$ do // For all genotypes
11                 $\langle D_{ji} \rangle^Y$.append $(\Delta(\langle g^{j,\lambda} \rangle^Y, \langle g^{i,\lambda} \rangle^Y))$
12         for $\forall i \neq j$ do // For all (unordered) pairs
13             if $j < i$ then
14                 $\langle d \rangle^Y \leftarrow$ HW($\langle D_{ji} \rangle^Y$)
15             else
16                 $\langle d \rangle^Y \leftarrow$ HW($\langle D_{ij} \rangle^Y$)
17             if $\langle \alpha^j \rangle^Y == \langle \alpha^i \rangle^Y$ then // If records have same label
18                 $\langle m^{\text{hit}}_{\text{dist}} \rangle^Y, \langle m^{\text{hit}}_{\text{ineq}} \rangle^Y \leftarrow$ kNN($\langle m^{\text{hit}}_{\text{dist}} \rangle^Y, \langle m^{\text{hit}}_{\text{ineq}} \rangle^Y, \langle d \rangle^Y, k$)
19             else
20                 $ysm^{\text{miss}}_{\text{dist}}, \langle m^{\text{miss}}_{\text{ineq}} \rangle^Y \leftarrow$ kNN($\langle m^{\text{miss}}_{\text{dist}} \rangle^Y, \langle m^{\text{miss}}_{\text{ineq}} \rangle^Y, \langle d \rangle^Y, k$)
21         $\langle W \rangle^Y \leftarrow \langle W \rangle^Y + \langle m^{\text{miss}}_{\text{ineq}} \rangle^Y - \langle m^{\text{hit}}_{\text{ineq}} \rangle^Y$
22     for $\forall j$ do
           // The features are sorted by weight and only the first (best)
           // $\varphi \cdot a$ are retained
23         $\langle g'^j \rangle^Y \leftarrow$ kNN($\langle g^j \rangle^Y, \langle W \rangle^Y, \varphi \cdot a$)
24         $\langle r'^j \rangle^Y \leftarrow (\langle g'^j \rangle^Y [1 : \varphi \cdot a]), \langle \alpha^j \rangle^Y)$
25     $\langle R' \rangle^Y := \{\langle r'^1 \rangle^Y, \dots, \langle r'^k \rangle^Y\}$
26     return $\langle R' \rangle^Y$
```

Prot. 1: Private Relief-F protocol. PReliefF takes a dataset $\boldsymbol{R}$ and a ratio φ as input and returns the most weighted attributes. The referenced k-Nearest Neighbors function `kNN` is slightly adapted from [20].

used in precision estimation of the computed models are also aggregated into one value. Thus, it is beneficial to keep those operations in arithmetic sharing which allows to perform integer addition locally. However, arithmetic sharing is restricted to only additions and multiplications, and conversions to and from Boolean sharing may be more expensive than the evaluation of a purely Boolean circuit. In the following, we design *novel* efficient building blocks that improve over the PMDR protocol that uses only Boolean sharing by *two* orders of magnitude.

4.1 Secure Arithmetic Greater Than (AGT)

A Boolean Greater Than (GT) gate requires ℓ AND gates if optimized for AND size [25] and $3\ell - \lceil \log_2 \ell \rceil - 2$ if optimized for AND depth [39]. Moreover, the latter still has $\lceil \log_2 \ell \rceil + 1$ AND depth and incurs the corresponding number of communication rounds.

Baseline Arithmetic Greater Than Protocol. Here, we give a baseline Arithmetic Greater Than (AGT) protocol that compares two integers $x_0, x_1 \in \mathbb{Z}_{2^\ell} : x_0, x_1 < 2^{\ell_{in}}$ in arithmetic sharing, where $\ell_{in} = \ell - 1$. The protocol is as follows: (1) compute $\langle\delta\rangle^A \leftarrow \langle x_0\rangle^A - \langle x_1\rangle^A$, (2) decompose it to single bits as $\langle\delta\rangle^B \leftarrow \texttt{A2B}(\langle\delta\rangle^A)$ (cf. [11]), and (3) return the MSB of $\langle\delta\rangle^B$. It requires only sharing of 2ℓ bits — ℓ bits by the garbler and ℓ bits by the evaluator — and $\ell - 1$ AND gates in Yao sharing. It outputs a bit in Boolean sharing. Later in this section we show a protocol with even better communication and use this protocol as a baseline for comparison.

Our baseline protocol requires only one communication round and its only limitation is that the input values need to be smaller than $2^{\ell_{in}}$, where $\ell_{in} = \ell - 1$. It requires $\ell(4.5\kappa + 5) - 1.5\kappa - 5$ bits of communication in total: $3\ell\kappa$ for re-sharing both arithmetic shares in Yao sharing and $(\ell - 1) \cdot (1.5\kappa + 5)$ for computing the sum of both shares in a GC using three-halves garbling[37].

Our Novel AGT Construction with Low Communication. Here, we introduce a novel, alternative approach to compute the AGT gate with significantly lower communication inspired by [12] and [35]. The idea of our protocol is based on the fact that, in contrast to share reconstruction using an addition circuit, we only need to compute the MSB, but not the full addition circuit. To compute the MSB we need $\langle\delta\rangle_0[\ell]$, $\langle\delta\rangle_1[\ell]$ and the carry bit $\langle c\rangle[\ell]$, where the latter is computed from the previous bits in the shares. However, we can skip the computation of the intermediate carry bits and directly compute the MSB by utilizing 1-out-of-N OT [24] with less communication than using our baseline protocol shown above. A batch-mode extension is described in the full version of this paper.

Toy Example. Let ℓ be small, e.g., ℓ=4, and $\langle\delta\rangle_0^A, \langle\delta\rangle_1^A \in \mathbb{Z}_{2^\ell}$ are arithmetic shares of $\delta = x_1 - x_0 : x_0, x_1 < 2^{\ell-1}$. P_0 uses $\langle\delta\rangle_0^A$ as its choice index in OT [24]. P_1 samples a uniformly random mask bit $r \leftarrow\!\!\$ \{0, 1\}$ and generates messages $\{(i + \langle\delta\rangle_1 \mod 2^\ell > 2^{\ell-1} - 1) \oplus r\}_{i=0}^{2^\ell-1}$. Then, P_0 obtains and sets $\langle \mathrm{MSB}\rangle_0^B := (\langle\delta\rangle_0^A + \langle\delta\rangle_1^A \mod 2^\ell > 2^{\ell-1} - 1) \oplus r$, and P_1 sets $\langle \mathrm{MSB}\rangle_1^B := r$. The communication complexity of this protocol is $2\kappa + 2^\ell$ bits, which equals 264 bits for bit length ℓ=3 and is 5.8× more efficient than the baseline protocol. The problem that arises here is that this toy protocol is not practical for large integers, e.g., the communication for ℓ=31 is 4.29 GB, which is orders of magnitude worse than our baseline.

Our Novel AGT for Integers of any Bit-Length. To reduce the communication for integers of arbitrary bit-length, we design an iterative approach that splits an integer into chunks and, in a nutshell, computes the carry bit for each of the intermediate chunks and extracts the MSB from the last chunk.

The best amortized per-bit communication in 1-out-of-N OT [24] is achieved with N=2^6 and equals $(2\kappa + 2^6)/6 = 53.3$ bits. Although N=2^7 requires 54.8 bits (amortized), it incurs less communication rounds in our AGT protocol. Our protocol consists of two subprotocols: (1) OT on the first chunk and (2) OT on

```
Function PMDR($\boldsymbol{R_i}$):
    Each data owner $P_i$ locally randomly permutes its raw dataset $\boldsymbol{R}_i := (r_i^1, \ldots, r_i^{k_i})$, where the
      record $r_i^j := ((g_i^{j,1}, \ldots, g_i^{j,a_i}), \alpha_i^j) : r_i^j \in \boldsymbol{R}_i$ with each genotype $g^{j,\lambda} \in \{1,2,3\}$ of person $j$ at
      locus $\lambda$ and each group $\alpha \in \{+,-\}$, denotes the case and control group, respectively, and
      splits it into $s$ equal parts $\boldsymbol{R}_i^1, \ldots, \boldsymbol{R}_i^s$.
    for $j = 1 \ldots s$ do // For each of $s$ cross-validation steps
        for $i = 1 \ldots N$ do // Each party splits its dataset for cross-validation.
            $\boldsymbol{R}_i^{\mathrm{val}} := \{r_i^j\}_{j=|\boldsymbol{R}_i|\cdot j/s}^{|\boldsymbol{R}_i|\cdot(j+1)/s}$
            $\boldsymbol{R}_i^{\mathrm{test}} := \boldsymbol{R}_i \setminus \boldsymbol{R}_i^{\mathrm{val}}$
        for $\lambda_1 \in [a]$ do // For each pair of loci $\lambda_1$ and $\lambda_2$
            for $\lambda_2 \in [a] \setminus \{\lambda_1\}$ do
                // Each party locally counts the observed genotypes
                // for test ($\boldsymbol{T}$) and validation ($\boldsymbol{V}$) sets.
                for $i = 1 \ldots N$ do
                    $(\boldsymbol{T}_i^{\lambda_1,\lambda_2,+},\ \boldsymbol{T}_i^{\lambda_1,\lambda_2,-},\ \boldsymbol{V}_i^{\lambda_1,\lambda_2,+},\ \boldsymbol{V}_i^{\lambda_1,\lambda_2,-}) \leftarrow \mathtt{Count}(\boldsymbol{R}_i^{\mathrm{test}}, \boldsymbol{R}_i^{\mathrm{val}}, \lambda_1, \lambda_2)$
                // All parties share and aggregate their counts using Arithmetic
                // sharing.  Remark:  Sharing is done locally using a PRG.
                $\langle \boldsymbol{X}^{\lambda_1,\lambda_2,g}\rangle^A \leftarrow \sum_{i=1}^N \langle \boldsymbol{X}_i^{\lambda_1,\lambda_2,g}\rangle^A$ for $\boldsymbol{X} \in \{\boldsymbol{T}, \boldsymbol{V}\}$ and $g \in \{+,-\}$
                // Compute the high risk prediction model as Boolean matrix $\boldsymbol{H}^{\lambda_1,\lambda_2}$.
                // If #cases/#controls is greater than a public threshold $t_h = t_h^+/t_h^-$,
                // the cell that corresponds to the genotype combination $(i,j)$ is
                // marked as high risk, indicated with $\langle 1\rangle^B$.
                for $i, j \in \{1,2,3\}$ do // For each combination of genotypes
                    // This is equivalent to computing $(\boldsymbol{T}^{\lambda_1,\lambda_2,+}[i,j]/\boldsymbol{T}^{\lambda_1,\lambda_2,-}[i,j]) > t_h$.
                    $\langle \mathit{num\_cases}\rangle^A \leftarrow t_h^+ \cdot \langle \boldsymbol{T}^{\lambda_1,\lambda_2,+}[i,j]\rangle^A$
                    $\langle \mathit{num\_controls}\rangle^A \leftarrow t_h^- \cdot \langle \boldsymbol{T}^{\lambda_1,\lambda_2,-}[i,j]\rangle^A$
                    // Mark this cell as high risk if #cases/#controls $>$ $t_h$.
                    $\langle \boldsymbol{H}^{\lambda_1,\lambda_2}[i,j]\rangle^B \leftarrow \mathtt{AGT}(\langle \mathit{num\_cases}\rangle^A, \langle \mathit{num\_controls}\rangle^A)$
                    // Swap validation counts if the current cell is high risk.
                    $\mathsf{ASWAP}(\langle \boldsymbol{H}^{\lambda_1,\lambda_2}[i,j]\rangle^B, \langle \boldsymbol{V}^{\lambda_1,\lambda_2,+}[i,j]\rangle^A, \langle \boldsymbol{V}^{\lambda_1,\lambda_2,-}[i,j]\rangle^A)$
                // Compute number of correcly and incorrectly classified samples.
                $\langle \mathit{num\_correct}\rangle^A \leftarrow \sum_{i,j} \langle \boldsymbol{V}^{\lambda_1,\lambda_2,-}[i,j]\rangle^A$ for $i, j \in \{1,2,3\}$
                $\langle \mathit{num\_wrong}\rangle^A \leftarrow \sum_{i,j} \langle \boldsymbol{V}^{\lambda_1,\lambda_2,+}[i,j]\rangle^A$ for $i, j \in \{1,2,3\}$
                // Store a bit indicating good/bad accuracy given a public accuracy
                // threshold $t_a = t_a^+/t_a^-$.
                $\boldsymbol{A}^j[\lambda_1, \lambda_2] \leftarrow \mathtt{AGT}(t_a^+ \cdot \langle \mathit{num\_correct}\rangle^A, t_a^- \cdot \langle \mathit{num\_wrong}\rangle^A)$
    for $\lambda_1 \in [a]$ do // For each pair of loci $\lambda_1$ and $\lambda_2$
        for $\lambda_2 \in [a] \setminus \{\lambda_1\}$ do
            // Output 1 if at least one of the cross validation steps $\lambda_1$ and $\lambda_2$
            // were marked as high risk.
            $\boldsymbol{O}[\lambda_1, \lambda_2] = \bigvee_{j=1}^s \boldsymbol{A}^j[\lambda_1, \lambda_2]$
    return $\boldsymbol{O}$
```

Prot. 2: Arithmetic Private Multifactor Dimensionality Reduction (PMDR^{A+}) protocol for *two* loci (for simplicity). Notation: N denotes #parties, a denotes #loci. PMDR outputs all loci combinations with MDR models that have accuracy greater than a threshold t_a. The functionality `Count` counts genotypes belonging to cases and controls for the test and validation set.

the intermediate chunks and carry bits. We XOR the last computed carry bit in Boolean sharing with the MSBs of the shares of δ, which yields the shared comparison result.

The first subprotocol requires $(2\kappa + 2^{\ell_s})$ bits. The second subprotocol requires $\gamma(2\kappa + 2^{\ell_s}) + 2\kappa + 2^{\epsilon}$ bits, where $\gamma = \lceil(\ell - \ell_s)/(\ell_s - 1)\rceil - 1$ is the number of the intermediate chunks and $\epsilon = \ell - \ell_s - 1 \bmod (\ell_s - 1)$ corresponds to the size of the remainder. For $\ell \geq \ell_s$ the total communication is equal to $(\gamma + 1)(2\kappa + 2^{\ell_s}) + \lceil\epsilon/(\ell_s - 1)\rceil(2\kappa + 2^{\epsilon})$ bits and the number of communication rounds is $\gamma + \lceil\epsilon/(\ell_s - 1)\rceil + 2$ due to sequential calls to the OT functionality. More concretely, for ℓ_s=7 and N=2^{ℓ_s} this translates to 384 bits and 2 rounds for ℓ_{in}=7, 1,028 bits and 4 rounds for ℓ_{in}=15, 1,920 bits and 6 rounds for ℓ_{in}=31, and 4,100 bits and 12 rounds for ℓ_{in}=63. Note that ℓ_{in} denotes the maximum bit-length of the integers, shared in $\mathbb{Z}_{2^{\ell+1}}$.

```
Function AGT($\langle x_0\rangle^A, \langle x_1\rangle^A, \ell_s$):
    // $\langle x_0\rangle^A, \langle x_1\rangle^A$ are secret-shared
    // in $\mathbb{Z}_{2^\ell}$ with $x_0, x_1 < 2^{\ell-1}$
    $\langle\delta\rangle^A = \langle x_1\rangle^A - \langle x_0\rangle^A$
    $sel \leftarrow \langle\delta\rangle_0^A[1 : \ell_s]$
    $M \leftarrow (j + \langle\delta\rangle_1^A[1 : \ell_s] > 2^{\ell_s})_{j=1}^{2^\ell_s}$
    $r \leftarrow\$ \{0,1\}$
    $c \leftarrow \binom{N}{1}\text{-OT}(sel, \{m \oplus r\}_{m\in M})$
    // Counter for the previous chunk
    $\ell_{\text{prev}} \leftarrow \ell_s + 1$
    while $\ell_{prev} < \ell$ do
        $\ell'_s \leftarrow \min(\ell_s - 1, \ell - \ell_{\text{prev}})$
        $\ell_{next} \leftarrow \ell_{\text{prev}} + \ell'_s - 1$
        $sel \leftarrow \langle\delta\rangle_0^A[\ell_{\text{prev}} : \ell_{next}]$
        $sel \leftarrow sel + c \cdot 2^{\ell'_s}$
        $\langle\delta'\rangle_1^A \leftarrow \langle\delta\rangle_1^A[\ell_{\text{prev}} : \ell_{next}]$
        $M_0 \leftarrow \{j + \langle\delta'\rangle_1^A > 2^{\ell'_s}\}_{j=1}^{2^{\ell'_s}}$
        $M_1 \leftarrow \{j + \langle\delta'\rangle_1^A + 1 > 2^{\ell'_s}\}_{j=1}^{2^{\ell'_s}}$
        $M \leftarrow M_r \cup M_{1-r}$
        $r \leftarrow\$ \{0,1\}$
        $c \leftarrow \binom{N}{1}\text{-OT}(sel, \{m \oplus r\}_{m\in M})$
        $\ell_{\text{prev}} \leftarrow \ell_{\text{next}} + 1$
    $\langle b\rangle^B = (\langle b\rangle_0^B, \langle b\rangle_1^B) :=$
      $(c \oplus \langle\delta\rangle_0^A[\ell], r \oplus \langle\delta\rangle_1^A[\ell])$
    return $\langle b\rangle^B$
```

Prot. 3: Our optimized Arithmetic Greater Than (AGT) protocol. The substring bit-length is denoted by ℓ_s.

To the best of our knowledge, the only secure comparison protocol of additively secret-shared integers was recently introduced by Rathee et al. [35, Algorithm 1] and showed to be more efficient than the comparison protocols of XOR-shared integers [10]. The difference to our protocol is that their protocol securely compares two cleartext integers, x and y, and produces a secret-shared result. Inspired by their construction, we extend their protocol for comparing $\langle x\rangle^A > \langle y\rangle^A$ by restricting $x, y < 2^{\ell_{in}}$ and computing the comparison as $\langle x\rangle^A - \langle y\rangle^A < 2^{\ell_{in}}$, thus "sacrificing" one bit for the comparison result. Since the subtraction can be done locally, our protocol can be seen as MSB extraction from a secret-shared integer, which corresponds to [35, Algorithm 2] which is, in turn, based on [35, Algorithm 1].

We provide a more communication-efficient construction compared to [35, Algorithm 2] that requires only $\binom{N}{1}$-OT invocations and no computation of AND gates. For ℓ_{in}=32-bit inputs, our protocol for MSB extraction requires 1.5× less communication (our 1,920 bits vs. their 2,914 bits), but one more communication round (our 6 rounds vs. their 5 rounds). The MSB extraction from ℓ=32-bit integers can be used to realize comparison of ℓ_{in}=31-bit integers.

Security. Informally, our AGT protocol only makes multiple consecutive calls to the $\binom{N}{1}$-OT functionality in a black-box way, and it produces uniformly distributed outputs in each step. Concretely, the first call to the $\binom{N}{1}$-OT functionality takes in the first ℓ_s bits of $\langle\delta\rangle^A$ and produces a secret share $(c, r) \in \{0,1\}^2$, where

$c := (\langle\delta\rangle_0^A[1 : \ell_s] + \langle\delta\rangle_1^A[1 : \ell_s] \geq 2^{\ell_s}) \oplus r$ and r is a random bit generated and known only by the OT sender. Since r is uniformly distributed and c is "masked" by r, the output is uniformly distributed and thus (c, r) is a secret share. The further calls to $\binom{N}{1}$-OT are invoked on the remaining substrings of $\langle\delta\rangle^A$. Namely, P_0 and P_1 call $\binom{N}{1}$-OT, which produces a new (c, r) pair, where $c := (\langle\delta\rangle_0^A[\ell_{\text{prev}} : \ell_{\text{prev}} + \ell'_s - 1] + \langle\delta\rangle_1^A[\ell_{\text{prev}} : \ell_{\text{prev}} + \ell'_s - 1] + (c_{\text{prev}} \oplus r_{\text{prev}}) \geq 2^{\ell'_s}) \oplus r$, where $(c_{\text{prev}}, r_{\text{prev}})$ are the results of the previous $\binom{N}{1}$-OT call (for better readability), and r is again a random bit generated and known only by the OT sender. As in the first step, the result is a secret share. The final result is computed locally on the available secret shares. It is easy to see that the result is also a secret share. A formal security proof for our AGT protocol can trivially be derived from the security proof of [35, Algorithm 2].

4.2 Secure Arithmetic Swap (ASWAP)

Another important building block in our PMDR protocol is Secure Arithmetic Swap (ASWAP), which obliviously swaps arithmetic inputs. More formally, ASWAP takes in a secret-shared bit $\langle b\rangle^B$ and a pair of additively shared integers $(\langle x_0\rangle^A, \langle x_1\rangle^A)$, and it outputs $(\langle x'_0\rangle^A, \langle x'_1\rangle^A) := (\langle x_b\rangle^A, \langle x_{1-b}\rangle^A)$.

A straightforward realization of ASWAP uses four multiplication gates for computing

$$i \in \{0, 1\} : \langle x'_i\rangle^A := (\neg\langle b\rangle^B \cdot \langle x_i\rangle^A + \langle b\rangle^B \cdot \langle x_{1-i}\rangle^A).$$

Note that the secure multiplication $\langle b\rangle^B \cdot \langle x\rangle^A$ can be realized using just two additively correlated OTs (cf. [1]) as described in [38]. This protocol requires $8(\kappa + \ell)$ bits of communication in total.

In the following, we design an ASWAP protocol that requires only *one* multiplication, and consequently $2(\kappa + \ell)$ bits of communication, and thus yields a factor 4 communication improvement compared to the naïve protocol. To construct our efficient protocol for ASWAP, we take inspiration from the *Boolean* swap protocol (called "X gate" in their work) by Kolesnikov and Schneider [26], which requires only one AND gate to perform an oblivious swap conditioned on $\langle b\rangle^B$ and can be seen as a special case of ASWAP for integers of bit length ℓ=1. Unfortunately, their protocol is not trivially generalizable to ASWAP for integers in $\mathbb{Z}_\ell$ with $\ell > 1$ because it relies on XOR, which is not trivially realizable on arithmetic shares. Our ASWAP protocol is depicted in Prot. 4. As for AGT, the batch-mode extension of this building block is described in the full version of this paper.

Beyond being useful for PMDR, our ASWAP protocol is of independent interest, e.g., combined with our AGT protocols, we can efficiently sort arithmetic values, i.e., using sorting networks on arithmetic circuits. This may be very beneficial in scenarios where the inputs to the sorting network are aggregated, since the addition operation is local in arithmetic sharing but costs $\ell - 1$ AND gates in a Boolean circuit [39]. Also, this omits expensive conversions (cf. [11]) if the further circuit is arithmetic, e.g., for efficient multiplications in $\mathbb{Z}_{2^\ell}$.

Although our ASWAP protocol is admittedly not complex, it has, to the best of our knowledge, never been used in the literature. We believe that the reason is that it has only recently been shown how to compute $\langle b \rangle^B \cdot \langle x \rangle^A$ efficiently [38].

```
Function ASWAP($\langle b \rangle^B, \langle x_0 \rangle^A, \langle x_1 \rangle^A$):
    $\langle \delta \rangle^A \leftarrow \langle b \rangle^B \cdot (\langle x_1 \rangle^A - \langle x_0 \rangle^A)$
    $\langle x_0' \rangle^A \leftarrow \langle x_0 \rangle^A + \langle \delta \rangle^A$
    $\langle x_1' \rangle^A \leftarrow \langle x_1 \rangle^A - \langle \delta \rangle^A$
    return $(\langle x_0' \rangle^A, \langle x_1' \rangle^A)$
```

Prot. 4: Secure Arithmetic Swap (ASWAP) protocol. Note that addition and subtraction are free in arithmetic sharing.

Security. Since both ASWAP and batch-ASWAP operate only on unmodified secret shares and use the well-known correlated OT technique [1] to produce the output secret shares in a black-box way, the security proof for both our primitives is trivial.

4.3 Communication of PMDR

Here, we evaluate the communication improvement gained by using our optimizations for PMDR. Our bottom line is a one-to-one translation of the PMDR algorithm (see Fig. 1) to a Boolean circuit (PMDR^Y that is evaluated in Yao sharing completely, which is often a very efficient solution due to the constant number of communication rounds in Yao sharing.

Our optimization of the PMDR protocol using our novel, more efficient arithmetic building blocks is denoted as PMDR^{A+}, which keeps data in arithmetic sharing, thus avoiding the costly conversions between different representations, and performs only very few operations in Boolean sharing. For the concrete communication costs of the gates, we refer the reader to [39] and [11] for arithmetic and Boolean sharing, and to [37] for Yao sharing. In the following, we fix the bit length of the integers to ℓ=32, which allows for up to 2^{31} genome samples in total with the standard threshold parameters. We, conventionally, always perform s=10 cross-validation steps.

PMDR^Y. For each combination of L loci and each of 3^L possible combination of alleles, this protocol requires (1) $N-1$ additions for aggregation of allele counts, (2) two multiplications and one comparison for determining the risk category, and (3) one swap operation [26] to set low and high risk counts in the validation set. Afterwards, to determine the accuracy of the model, the validation counts are summed up, which requires $2 \cdot 3^L - 1$ additions, two multiplications and one comparison. For the costs of these operations, see [39]. Finally, $s-1$ AND gates in Boolean sharing are used to compute a secret-shared bit that indicates whether the model was accurate in at least one cross-validation step. In total, for each combination of loci the protocols requires

$$2s(3^L(8\ell^2 + s - 1) - \ell(4\ell - 1)) - s + 1$$

AND gates. This corresponds to 1 394 891 AND gates or 34.34 MB of communication for L=2 loci, and 4 347 251 AND gates or 101.05 MB of communication for L=3 loci, using three-halves garbling with $1.5\kappa + 5$ bits per AND gate [37].

$PMDR^{A+}$. For each combination of L loci and each of 3^L possible combination of alleles, our PMDR^{A+} protocol requires one AGT and one ASWAP gate. Then, for each combination of L loci, another AGT gate is required. And, finally, $s-1$ AND gates in Boolean sharing are needed to compute the secret-shared interaction indication. *All* other operations in this protocol are *non-interactive*.

The total communication translates to $s(3^L(\ell(4\kappa+1)+2(\kappa+\ell))+\ell(4\kappa+1))+s-1$ bits, which equals only 208.8 kB of communication for L=2 loci and 585.3 kB for L=3 loci. Compared to PMDR^{Y}, this yields an improvement by a factor of 164× for L=2 and by a factor of 172× for L=3.

5 Implementation

We implement our protocols for Private Epistasis Analysis (PEA)[6] using the MOTION framework [6] for Secure Multi-Party Computation (MPC). The reason for choosing MOTION is its efficiency and flexibility. Due to the number of new building blocks that we constructed and/or implemented, e.g., the three-halves garbling [37] and our new AGT protocol (cf. Prot. 3), we required an MPC framework that admits changes in its internal infrastructure and protocols with only moderate implementation overhead. Another selection criterion was the efficiency of the framework. MOTION satisfies both requirements. We detail and analyze our implementation of three-halves garbling in App. B.

6 Evaluation

We evaluate PEA on two servers equipped with Intel Core i9-7960X processors and 128 GB of RAM. We average all our benchmarks over 10 runs.

We use synthetic data as the input in our benchmarks due to two reasons. (1) MPC is input-oblivious by its security definition and thus the performance of our protocols is input-independent. (2) Since our protocols are fully accurate, we can discern no useful insight by using real (and hard to get access to) privacy-sensitive datasets, and thus, we favor the ethically better decision to use the least privacy-intrusive data source, i.e., synthetic data.

Settings. We evaluate two settings for Private Epistasis Analysis (PEA):
WAN. Two medical institutions perform PEA directly, aggregating their own databases in MPC. Our benchmarking environment naturally resembles the scenario where the two medical institutions are located very close to each other. However, our PEA protocols are either constant-round or are highly parallelized, so the most important performance aspect is the network bandwidth. We expect the medical institutions to have a high-bandwidth Internet connection. In our benchmarking environment, we use a 10 Gbit/s bandwidth network connection but conservatively restrict the latency to 50 ms using the `tc` tool[7] to simulate the WAN setting.

[6] https://encrypto.de/code/EPISTASIS
[7] https://man7.org/linux/man-pages/man8/tc.8.html

LAN. Several medical institutions send their *secret shared* data to outsourcing servers [21] that aggregate the received data and compute PEA on the aggregated data and finally send back the *shared* result. The outsourcing servers cannot infer any information about the input and output data as well as the intermediate values, but they are assumed not to collude. Such servers may be two cloud computing providers located close to each other, e.g., near the same Internet exchange point, thus having a high-bandwidth, low-latency connection. Thus, in the LAN setting we do not put additional constraints on the network and use a network connection with 10 Gbit/s throughput and 0.2 ms latency.

6.1 Performance of PReliefF and PTuRF

The results of our performance benchmarks of the private feature selection algorithms are shown in Tab. 1.

During our evaluation, RAM utilization was the a bottleneck during feature selection. This is not surprising, as our implementation was not optimized for space efficiency, but runtime and communication efficiency. Because of RAM exhaustion, PTuRF could not be benchmarked across the full parameter space.

As expected, a linear growth pattern, after a steep initial increase, can be observed in the runtimes of PReliefF and PTuRF. Due to the constant number of interaction rounds in Yao's GC protocol, the additional latency in the WAN setting has no strong effect on the measured runtimes.

Although the number of features to consider is reduced in every iteration, PTuRF's higher sorting work load leads to worse performance compared to the simpler PReliefF algorithm. However, the pruning of noisy features is shown in [31] to increase the robustness of the results.

Due to its linear runtime–size complexity, it is practical to perform PReliefF on datasets with real-world sizes (e.g., ≈ 100 records with $\approx 10{,}000$ features [7]) in less than a day.

Tab. 1: Runtimes and communication for our private ReliefF (PReliefF^Y) and private TuRF (PTuRF^Y) protocols filtering $|R|$ records with 10 SPNs each.

			$\lvert R\rvert$=4	$\lvert R\rvert$=8	$\lvert R\rvert$=20	$\lvert R\rvert$=40	$\lvert R\rvert$=60	$\lvert R\rvert$=80	$\lvert R\rvert$=100
PReliefF^Y	Runtime	LAN	1.00 s	1.74 s	5.13 s	13.15 s	21.19 s	33.52 s	50.14 s
		WAN	1.98 s	2.30 s	7.99 s	15.04 s	23.68 s	36.21 s	52.29 s
	Comm.		3.75 MB	9.63 MB	40.98 MB	138.93 MB	294.13 MB	506.45 MB	775.93 MB
PTuRF^Y	Runtime	LAN	1.09 s	2.13 s	13.65 s	83.37 s	—	—	—
		WAN	1.49 s	2.51 s	14.34 s	85.54 s	—	—	—
	Comm.		4.11 MB	11.13 MB	107.15 MB	510.77 MB	—	—	—

6.2 Performance of PMDR

The performance of PEA's Private Multifactor Dimensionality Reduction (PMDR) is reported in Tab. 2. The exponential scaling in the number of interacting loci

L is clearly visible, both in the runtime and the communication, which for L=3, N=1,000 reaches nearly 100 TB.

However, for smaller numbers of SNPs a or lower interaction depths, such as a=1,000, L=2 or a=100, L=3, our PMDR implementation runs in less than an hour in both LAN and WAN.

The very low communication overhead for outsourcing leads to a practical solution for pooling and analyzing multiple institutions' genomic data.

6.3 Total Performance

Due to the exponential complexity of (P)MDR, combination with a preceding feature selection algorithm is a sensible practice. At the cost of leaking the number of filtered features, the reduced number of features significantly improves the efficiency and the result is more robust against noisy attributes. As described by Moore and White [31], it is hard to give a general estimate on feasible feature reduction, as the resulting accuracy is depending, among other factors, on the amount of noise, the size of the data set, and the heritability of the trait. However, they measure 80 % accuracy while using TuRF to remove 950 out of 1,000 features.

Tab. 2: Runtimes and communication for PEA's Arithmetic Private Multifactor Dimensionality Reduction (PMDR^{A+}) protocol with s=10 cross-validation steps using an interaction depth of L and a attributes.

	depth	a=10	a=100	a=1,000
LAN	L=2	1.71 s	24.85 s	43.08 min
	L=3	3.29 s	33.68 min	—
WAN	L=2	10.88 s	42.71 s	1.04 h
	L=3	10.01 s	47.71 min	—
Comm.	L=2	9.39 MB	1.03 GB	104.29 GB
	L=3	70.23 MB	94.64 GB	97.25 TB

As the performance benchmarks in the previous section show, this composition of both algorithms achieves only a performance gain for large numbers of features or for interaction depths larger than L=2. In those cases PMDR on itself becomes prohibitively long running and the reduction of the number of features by 10 % corresponds to a significant performance gain, e.g., 20 % improvement for a=10,000 features (40 instead of 50 million considered combinations).

Acknowledgments

This project received funding from the European Research Council (ERC) under the European Union's Horizon2020 research innovation program (grant agreement No.850990 PSOTI), and the German Ministry of Education and Research through the project HiGHmed (funding #01ZZ1802G). It was co-funded by the Deutsche Forschungsgemeinschaft(DFG) within SFB1119 CROSSING/236615297 and GRK2050 Privacy & Trust/251805230, and by the German Federal Ministry of Education and Research and the Hessen State Ministry for Higher Education, Research and the Arts within ATHENE.

Bibliography

[1] Asharov, G., Lindell, Y., Schneider, T., Zohner, M.: More efficient oblivious transfer extensions. JoC (2017)

[2] Barbujani, G., Colonna, V.: Human genome diversity: Frequently asked questions. Trends in Genetics (2010)

[3] Beaver, D.: Correlated pseudorandomness and the complexity of private computations. In: STOC (1996)

[4] Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Rindal, P., Scholl, P.: Efficient two-round OT extension and silent non-interactive secure computation. In: CCS (2019)

[5] Braun, L., Cammarota, R., Schneider, T.: A generic hybrid 2PC framework with application to private inference of unmodified neural networks. In: NeurIPS Workshop Privacy in Machine Learning (2021)

[6] Braun, L., Demmler, D., Schneider, T., Tkachenko, O.: MOTION - A framework for mixed-protocol multi-party computation. TOPS (2022)

[7] Chen, Q., Zhang, X., Zhang, R.: Privacy-preserving decision tree for epistasis detection. Cybersecurity (2019)

[8] Cho, H., Wu, D.J., Berger, B.: Secure genome-wide association analysis using multiparty computation. Nature Biotechnology (2018)

[9] Cho, Y.M., Ritchie, M.D., Moore, J.H., Park, J.Y., Lee, K.U., Shin, H.D., Lee, H.K., Park, K.S.: Multifactor-dimensionality reduction shows a two-locus interaction associated with Type 2 diabetes mellitus. Diabetologia (2004)

[10] Couteau, G.: New protocols for secure equality test and comparison. In: CANS (2018)

[11] Demmler, D., Schneider, T., Zohner, M.: ABY - a framework for efficient mixed-protocol secure two-party computation. In: NDSS (2015)

[12] Dessouky, G., Koushanfar, F., Sadeghi, A.R., Schneider, T., Zeitouni, S., Zohner, M.: Pushing the communication barrier in secure computation using lookup tables. In: NDSS (2017)

[13] Duncan, G.: Statistical confidentiality: principles and practice. Springer (2011)

[14] Dwork, C.: Differential privacy. In: ICALP (2006)

[15] Gilboa, N.: Two party RSA key generation. In: CRYPTO (1999)

[16] Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game. In: STOC (1987)

[17] Hahn, L.W., Ritchie, M.D., Moore, J.H.: Multifactor dimensionality reduction software for detecting gene–gene and gene–environment interactions. Bioinformatics (2003)

[18] Hamacher, K.: PETS Genome Privacy Workshop (2014)

[19] Ishai, Y., Kilian, J., Nissim, K., Petrank, E.: Extending oblivious transfers efficiently. In: CRYPTO (2003)

[20] Jarvinen, K., Leppakoski, H., Lohan, E.S., Richter, P., Schneider, T., Tkachenko, O., Yang, Z.: PILOT: Practical privacy-preserving Indoor Localization using OuTsourcing. In: EuroS&P (2019)

[21] Kamara, S., Raykova, M.: Secure outsourced computation in a multi-tenant cloud. In: IBM Workshop on Cryptography and Security in Clouds (2011)

[22] Kim, Y., Park, T.: Robust gene-gene interaction analysis in genome wide association studies. PloS One (2015)

[23] Kira, K., Rendell, L.A.: A practical approach to feature selection. In: Machine Learning (1992)

[24] Kolesnikov, V., Kumaresan, R.: Improved OT extension for transferring short secrets. In: CRYPTO (2013)

[25] Kolesnikov, V., Sadeghi, A.R., Schneider, T.: Improved garbled circuit building blocks and applications to auctions and computing minima. In: CANS (2009)

[26] Kolesnikov, V., Schneider, T.: Improved garbled circuit: Free XOR gates and applications. In: ICALP (2008)

[27] Kononenko, I.: Estimating attributes: Analysis and extensions of RELIEF. In: ECML (1994)

[28] Le, T.T., Simmons, W.K., Misaki, M., Bodurka, J., White, B.C., Savitz, J., McKinney, B.A.: Differential privacy-based evaporative cooling feature selection and classification with Relief-F and random forests. Bioinformatics (2017)

[29] Lee, S., Son, D., Kim, Y., Yu, W., Park, T.: Unified Cox model based multifactor dimensionality reduction method for gene-gene interaction analysis of the survival phenotype. BioData Mining (2018)

[30] Meng, Y., Groth, S., Quinn, J.R., Bisognano, J., Wu, T.T.: An exploration of gene-gene interactions and their effects on hypertension. International Journal of Genomics (2017)

[31] Moore, J.H., White, B.C.: Tuning ReliefF for genome-wide genetic analysis. In: European Conference on Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics (2007)

[32] Naveed, M., Ayday, E., Clayton, E.W., Fellay, J., Gunter, C.A., Hubaux, J.P., Malin, B.A., Wang, X.: Privacy in the genomic era. ACM Computing Surveys (2015)

[33] Newton-Cheh, C., Johnson, T., Gateva, V., Tobin, M.D., Bochud, M., Coin, L., Najjar, S.S., Zhao, J.H., et al.: Genome-wide association study identifies eight loci associated with blood pressure. Nature Genetics (2009)

[34] Ozaki, K., Ohnishi, Y., Iida, A., Sekine, A., Yamada, R., Tsunoda, T., Sato, H., Sato, H., Hori, M., Nakamura, Y., Tanaka, T.: Functional SNPs in the lymphotoxin-αgene that are associated with susceptibility to myocardial infarction. Nature Genetics (2002)

[35] Rathee, D., Rathee, M., Kumar, N., Chandran, N., Gupta, D., Rastogi, A., Sharma, R.: Cryptflow2: Practical 2-party secure inference. In: CCS (2020)

[36] Ritchie, M.D., Hahn, L.W., Roodi, N., Bailey, L.R., Dupont, W.D., Parl, F.F., Moore, J.H.: Multifactor-dimensionality reduction reveals high-order interactions among estrogen-metabolism genes in sporadic breast cancer. The American Journal of Human Genetics (2001)

[37] Rosulek, M., Roy, L.: Three halves make a whole? Beating the half-gates lower bound for garbled circuits. In: CRYPTO (2021)
[38] Schneider, T., Tkachenko, O.: EPISODE: Efficient Privacy-preservIng Similar sequence queries on Outsourced genomic DatabasEs. In: ASIACCS (2019)
[39] Schneider, T., Zohner, M.: GMW vs. Yao? Efficient secure two-party computation with low depth circuits. In: FC (2013)
[40] Tkachenko, O., Weinert, C., Schneider, T., Hamacher, K.: Large-scale privacy-preserving statistical computations for distributed genome-wide association studies. In: ASIACCS (2018)
[41] Wang, M.H., Cordell, H.J., Van Steen, K.: Statistical methods for genome-wide association studies. Seminars in Cancer Biology (2019)
[42] Yang, K., Weng, C., Lan, X., Zhang, J., Wang, X.: Ferret: Fast extension for correlated OT with small communication. In: CCS (2020)
[43] Yang, L., Qu, B., Xia, X., Kuang, Y., Li, J., Fan, K., Guo, H., Zheng, H., Ma, Y.: Impact of interaction between the G870A and EFEMP1 gene polymorphism on glioma risk in chinese han population. Oncotarget (2017)
[44] Yao, A.C.: How to generate and exchange secrets. In: FOCS (1986)
[45] Zahur, S., Rosulek, M., Evans, D.: Two halves make a whole. In: EUROCRYPT (2015)
[46] Zhang, H., Zheng, W., Hua, L., Wang, Y., Li, J., Bai, H., Wang, S., Du, M., Ma, X., Xu, C., Li, X., Gong, B., Wang, Y.: Interaction between PPAR γ and SORL1 gene with Late-Onset Alzheimer's disease in Chinese Han Population. Oncotarget (2017)

Appendix

A Secure Multi-Party Computation

Secure Multi-Party Computation (MPC) are cryptographic protocols to compute a joint function over distributed, private data without the need of a trusted third party. In this work, we consider security against passive (a.k.a. semi-honest) adversaries, which strictly follow the protocol but try to learn more information.

Oblivious Transfer (OT). In OT [3] the sender inputs messages (m_0, m_1), and the receiver inputs a choice bit c. At the end of the protocol, the receiver obtains m_c but no information about m_{1-c}, and the sender does not obtain any information. OT can be instantiated very efficiently using mostly symmetric cryptography [19] and it admits optimizations for MPC [1]. OT can be generalized to N instead of two messages [24], where the receiver holds a choice index $c \in \mathbb{Z}_N$ and obtains m_c. Recently, a "silent" OT [4, 42] was introduced with significantly less communication at the cost of higher computation. The current silent OT schemes beat the textbook OT extension [19, 1] in terms of runtime in networks with limited bandwidth, which is less interesting in our scenario, where medical institutions performing large-scale MPC likely have a high-bandwidth connection.

Yao's Garbled Circuits (GCs). GCs were introduced in [44]. The state of the art [37] requires $1.5\kappa + 5$ bits per AND gate, where κ=128 is the symmetric security parameter. GCs operate on Boolean circuits and work by garbling the truth tables: a random symmetric encryption key is generated for every possible value on every wire. The output wire-keys of a gate are doubly encrypted using the corresponding combination of input wire keys. Only the garbler, i.e., the party preparing the GC, can connect the entries in the GC to "cleartext" values. Then, the garbler sends the GC and its input keys to the evaluator. The evaluator obliviously obtains its input keys using OT. The GC is then evaluated. We denote a bit b "shared" in a GC as $\langle b \rangle^Y$ and call this Yao sharing.

Goldreich-Micali-Wigderson (GMW). Like GCs, the GMW protocol [16], named after its inventors Goldreich, Micali and Wigderson, operates on Boolean circuits. It achieves its privacy guarantees by splitting every input bit b in two XOR-shares and letting party P_i hold share $\langle b \rangle_i^B$. These shares are constructed as $\langle b \rangle_0^B \leftarrow\!\!\$ \{0, 1\}$ and $\langle b \rangle_1^B \leftarrow b \oplus \langle b \rangle_0^B$ and reconstructed by XOR-ing both shares. XOR gates can be evaluated locally by XOR-ing both local shares and AND gates are evaluated interactively [1]. We denote this version of GMW as Boolean sharing. The GMW protocol can be extended to arithmetic circuits with elements in $\mathbb{Z}_{2^\ell}$. We denote this extension as Arithmetic sharing. Similarly to Boolean GMW, the shares are generated as $\langle x \rangle_0^A \leftarrow\!\!\$ \mathbb{Z}_n$ and $\langle x \rangle_0^A \leftarrow x - \langle x \rangle_1^A$. The addition can be performed locally and the multiplication requires interaction [15].

B Three Halves Make a Whole Garbling Implementation

In order to provide the best possible estimation of our PEA protocols' efficiency, we implement in MOTION [6] "three-halves garbling" (3HG) [37]. To the best of our knowledge, this is the first implementation of 3HG. Our optimized 3HG engine can garble 11.2 M/s and evaluate 27.5 M/s AND gates. Compared to "two-halves garbling" (2HG) [45] in MOTION by Braun et al. [5], 3HG is 4.7× slower in terms of garbling and 2.5× slower in terms of evaluation. This is also a more significant slowdown of garbling than the factor of 2.1× estimated in [37], based on the number of hash function calls. Our profiling indicates that the two main bottlenecks are the 1.5× higher number of AES invocations and the significantly higher degree of branching in 3HG compared to 2HG. Considering the garbling rate, we can saturate the 10 Gbit/s network channel with 5 threads. Furthermore, our benchmark for evaluating 512 AES circuits in parallel in a GC shows a 2.2× speedup compared to [5] (our 0.22 s vs. their 0.5 s). However, this result should be taken with a grain of salt, since [5] introduced significant changes to MOTION, which may have affected the runtimes.