Cell Systems

Methods in Brief

Privacy-preserving genotype imputation with fully homomorphic encryption

Highlights

- Genome privacy in genotype imputation can be achieved using homomorphic encryption
- Privacy-preserving genotype imputation can scale to real-world applications
- Further improvements in algorithm design will enable secure haplotype inference
- Secure haplotype inference can improve accuracy achieved in secure genotype imputation

Authors

Gamze Gürsoy, Eduardo Chielle, Charlotte M. Brannon, Michail Maniatakos, Mark Gerstein

Correspondence

michail.maniatakos@nyu.edu (M.M.), mark@gersteinlab.org (M.G.)

In brief

Gürsoy, Chielle et al. show that mathematically guaranteed privacy preservation in genotype imputation can be achieved using statistical inference with homomorphic encryption. This algorithm is accurate and scalable to real-world applications and can further be used in designing secure genotype imputation servers.

CellPress

Cell Systems

CellPress OPEN ACCESS

Methods in Brief

Privacy-preserving genotype imputation with fully homomorphic encryption

Gamze Gürsoy,[1,2,6] Eduardo Chielle,[3,6] Charlotte M. Brannon,[1,2] Michail Maniatakos,[3,*] and Mark Gerstein[1,2,4,5,7,*]
[1]Program in Computational Biology and Bioinformatics, Yale University, New Haven, CT 06520, USA
[2]Department of Molecular Biophysics and Biochemistry, Yale University, New Haven, CT 06520, USA
[3]Department of Electrical and Computer Engineering, New York University Abu Dhabi, Saadiyat Island, Abu Dhabi, UAE
[4]Department of Computer Science, Yale University, New Haven, CT 06520, USA
[5]Department of Statistics and Data Science, Yale University, New Haven, CT 06520, USA
[6]These authors contributed equally
[7]Lead contact
*Correspondence: michail.maniatakos@nyu.edu (M.M.), mark@gersteinlab.org (M.G.)
https://doi.org/10.1016/j.cels.2021.10.003

SUMMARY

Genotype imputation is the inference of unknown genotypes using known population structure observed in large genomic datasets; it can further our understanding of phenotype-genotype relationships and is useful for QTL mapping and GWASs. However, the compute-intensive nature of genotype imputation can overwhelm local servers for computation and storage. Hence, many researchers are moving toward using cloud services, raising privacy concerns. We address these concerns by developing an efficient, privacy-preserving algorithm called $p-Impute$. Our method uses homomorphic encryption, allowing calculations on ciphertext, thereby avoiding the decryption of private genotypes in the cloud. It is similar to k-nearest neighbor approaches, inferring missing genotypes in a genomic block based on the SNP genotypes of genetically related individuals in the same block. Our results demonstrate accuracy in agreement with the state-of-the-art plaintext solutions. Moreover, $p-Impute$ is scalable to real-world applications as its memory and time requirements increase linearly with the increasing number of samples. $p-Impute$ is freely available for download here: https://doi.org/10.5281/zenodo.5542001.

INTRODUCTION

The decreasing cost of DNA sequencing and the clinical importance of genomic characterization of individuals have resulted in an exponential increase of available human genetic data (Sboner et al., 2011). Such data have tremendous clinical value in terms of understanding and characterizing rare diseases and genotype-phenotype associations (Manolio, 2010; Rockman and Kruglyak, 2006). However, the large amount of data being collected and the complexity of genomic analyses can overwhelm the capacity of servers. In part, the issue has to do with the disk space required to store thousands of samples locally. Therefore, funding agencies and institutions have begun outsourcing the computations to cloud services. For example, the National Human Research Institute strongly suggests its awardees to use the Genomic Data Science Analysis, Visualization, and Informatics Lab-space (AnVIL) (National Human Genome Research Institute, 2021), which is a cloud-based infrastructure for genomic data access, sharing, and computing across large genomic and genomic-related datasets. These cloud-based services raise privacy issues when used for sharing patients' genomic data. With the increasing use of genetic information in new avenues, such as forensics, the need for privacy-preserving analysis methods is greater than ever.

Genotype imputation is the process of statistical inference of unknown genotypes in a genome using the correlation between the single-nucleotide polymorphism (SNP) sites observed in population-based haplotype structures. Genotype imputation has immense value in phenotype-genotype association studies, especially when there is a large number of study participants. This approach allows researchers to perform cost-effective genotyping methods, such as genotyping arrays or low-coverage whole-genome sequencing, and obtain missing or low-quality genotypes through statistical inference. As the input and output of genotype imputation methods are sets of participants' genotypes, performing this analysis in the cloud could have serious privacy implications, such as exposure of the sensitive data to untrusted third parties. However, the large amount of data required to boost statistical power makes local storage and analysis cost prohibitive. For example, online services, such as the Michigan Imputation Server, allow researchers to impute genotypes against large-scale reference panels based on thousands of genomes. This requires uploading highly sensitive genetic information from many query individuals to a server. Although the input and the output of these services can be encrypted, these servers cannot provide end-to-end encryption and the query data become vulnerable during computation. Therefore, privacy-preserving genotype imputation methods are needed that

This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/).

allow users to utilize cloud-based services while protecting privacy. Here, we present a privacy-preserving genotype imputation method called *p-Impute* based on homomorphic encryption (HE). $p - Impute$ allows users to perform genotype imputation on encrypted genotype data and returns encrypted genotype outputs, thereby removing the privacy concerns related to outsourcing cloud services.

HE is a form of encryption with the capability of computing on encrypted data without access to a secret key (Armknecht et al., 2015). HE schemes are classified based on the different kinds of computation that they can successfully perform on encrypted data. Common types of encryption schemes include partial, somewhat, leveled fully, and fully HE (Acar et al., 2018). Partial HE (PHE) supports the evaluation of circuits consisting of only one type of gate (e.g., addition or multiplication). Somewhat HE (SHE) supports the evaluation of two types of gates but only for a subset of circuits. Leveled fully homomorphic encryption (LFHE) supports the evaluation of arbitrary circuits composed of multiple types of gates of bounded depth. Fully HE (FHE) supports the evaluation of arbitrary circuits composed of multiple types of gates of unbounded depth. These limitations in the circuit size and gate type are due to the noise introduced to the ciphertexts, which is added during encryption to guarantee the security of the encryption system. FHE overcame these limitations by introducing bootstrapping, which is an expensive operation that reduces the noise of ciphertexts ((Gentry, 2010) Halevi and Shoup, 2015). Moreover, in HE, addition operations are fast and produce little noise, whereas multiplications are slow and produce more noise. Furthermore, in HE, control flow cannot depend on the data. For example, we cannot perform an *if* clause on encrypted data. Therefore, the program must run obliviously to its sensitive data, which leads to further performance degradation (Micciancio, 1997). Although HE has been used for the analysis of genomic and biomedical clinical data (Kim and Lauter, 2015; Kocabaş and Soyata, 2014, Bos et al., 2014; McLaren et al., 2016), it requires a fast and scalable implementation to be used for genotype imputation.

We designed *p-Impute*, a privacy-preserving statistical inference algorithm for performing genotype imputation, using the Brakerski/Fan-Vercauteren (BFV) encryption scheme (Fan and Vercauteren, 2012) provided by the Microsoft SEAL (SEAL, 2019) library. Although BFV is an FHE scheme, the SEAL implementation of BFV does not allow for bootsrapping; therefore, we use it as an SHE scheme, because the number of operations required in our algorithm is known *a priori*. We overcame the challenges related to performance overhead through algorithm optimization, batching, and thread-level parallelism. We used 2,000 fully characterized genomes from the 1,000 Genomes Project (The 1000 Genomes Project Consortium, 2015) to model the relationship between tag and target SNPs, and then tested the model on the remaining 500 fully characterized genomes from the 1,000 Genomes Project (The 1000 Genomes Project Consortium, 2015). We further evaluated the performance of our algorithm on various independent datasets, such as those from the Genotype-Tissue Expression (GTEx) (The GTEx Consortium, 2013), Avon Longitudinal Study of Parents and Children (ALSPAC) (The UK10K Consortium, 2015), and PsychENCODE (The PsychENCODE Consortium, 2018) projects. *p-Impute* defines blocks of genomic regions for each missing genotype and enumerates all possible genotype combinations of tag SNPs. It then statistically infers the missing genotypes based on all of the genotypes in these genomic regions of *M*-genetically related individuals. These *M* individuals are selected from a database with known genomes (e.g., the 1,000 Genomes Project) based on the similar principles of traditional k-nearest neighbors algorithms. We compared *p-Impute* results with results from the state-of-the-art non-encrypted counterparts, IMPUTE2 (Howie et al., 2012), Beagle (Browning and Browning, 2009), and Minimac (Das et al., 2016). We found that *p-Impute* offers reasonable overall genotype accuracy (94%) compared with its plaintext counterparts (98%). We further broke down our predictions into different minor allele frequency (MAF) and ancestry categories. We found that rare alleles with heterozygous genotypes are hardest to impute by *p-Impute*. We also found that the genotype accuracy of the imputed SNPs from African genomes is slightly lower compared with plaintext solutions. Overall, p-Impute provides a mathematically guaranteed and fast genotype imputation tool with a slight decrease in accuracy as a trade-off for its scalability and privacy guarantees.

RESULTS

p-Impute can be used in clinical and research settings

The motivations and logic for our genotype-imputation model can be summarized as follows: (1) Genotype imputation must be outsourced in the cloud because we cannot afford to download and compute thousands of genomes locally. (2) We must protect the private information, which are the input and output of the genotype imputation (SNPs). (3) Model privacy is needed in case the training data are not publicly available. Note that we built our model using the publicly available 1,000 Genomes dataset. (4) HE provides mathematically guaranteed privacy for the input and output of the genotype-imputation problem by converting them into ciphertext and operating in the encrypted domain.

In a real-world setting, our server provider is the cloud services with someone (let us say ‘‘Bob’’) who developed an imputation model that can operate on encrypted data and our client (let us say ‘‘Alice’’), who wants to impute genomes, is the researcher or clinician. Consider a scenario in which Alice has a set of genotyped tag SNPs and would like to impute the genotypes of her missing SNPs (target SNPs) using the tag SNPs. She would like to outsource the imputation to Bob, as Bob has a cloud-based genotype-imputation model. However, Alice does not trust Bob enough to send him her SNPs. Instead, Bob offers an imputation method in the encrypted space. Alice encrypts her tag SNP genotypes with her public key and sends them to Bob. Bob runs his imputation method, obtains encrypted results, and sends them to Alice. Alice uses her private key to decrypt the results, which are her missing SNP genotypes (Figure 1A).

p-Impute is a simple yet intuitive algorithm for genotype imputation

A phasing-free, fast plaintext solution as a base

Traditional plaintext genotype imputation methods, such as IMPUTE2 (Howie et al., 2012), Beagle (Browning and Browning, 2009), and Minimac (Das et al., 2016), first phase the genome into haplotypes. Next, they determine the best haplotype block

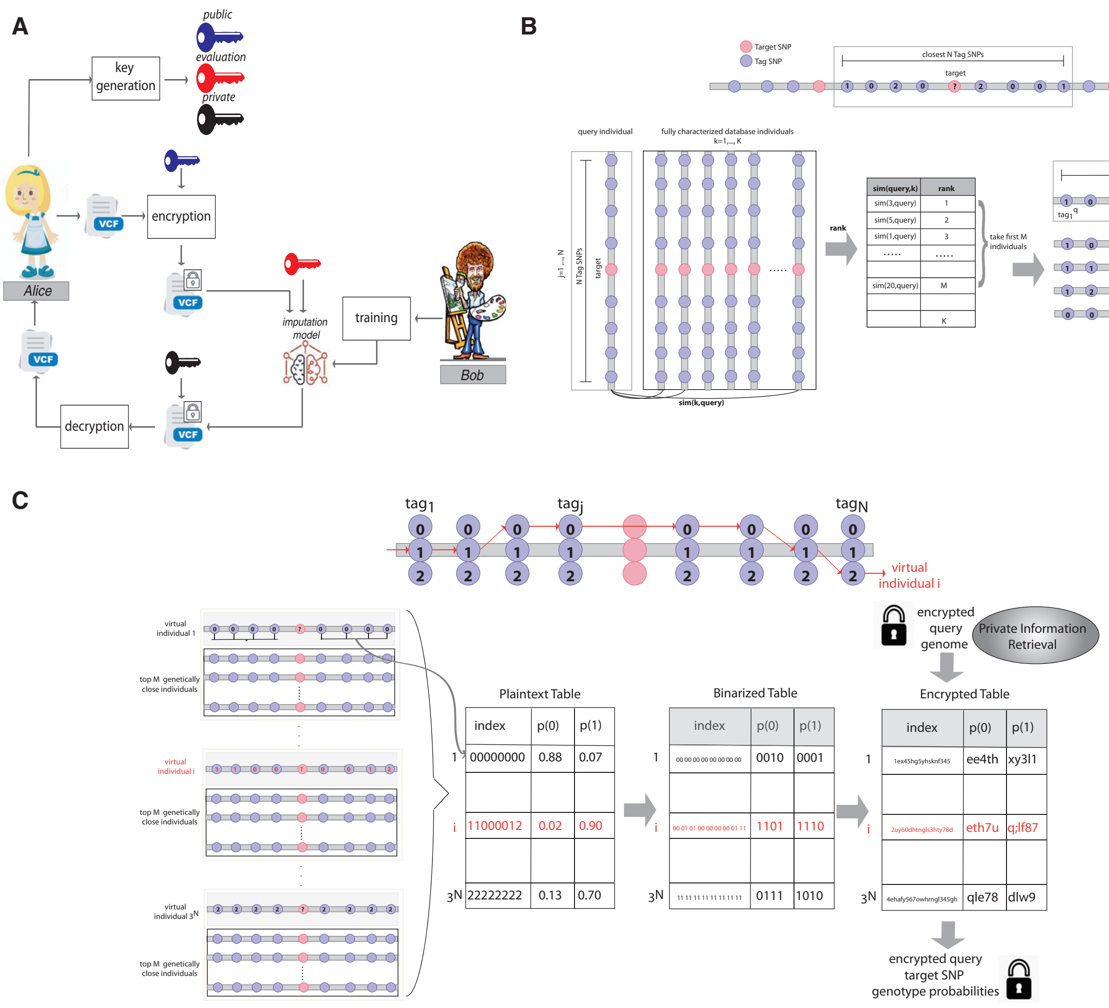

Figure 1. Threat Model and *p-Impute* algorithm in plaintext and encrypted domain

(A) Illustration of how genotype imputation with HE works in practice. After generating encryption (public), decryption (private), and evaluation keys, Alice encrypts the vcf file with her tag SNP genotypes with her encryption key and sends the encrypted genotypes and the evaluation key to Bob. Bob uses the encrypted genotypes as input to his privacy-preserving imputation model and outputs encrypted genotypes for Alice's missing SNPs and sends them to Alice. Alice can decrypt the output using her decryption key. Bob never receives any unencrypted data and never sees the plaintext results from his model. Bob can be thought of as the cloud service provider, such as AWS or Google cloud.

(B) After determining the neighboring tag SNPs for the missing genotype, the similarity between the query individual and each individual in the 1,000 Genomes dataset is calculated. These individuals are ranked based on the similarity to the query individual and the top *M* individuals are used to calculate the probability of each genotype being observed at the missing SNP location based on the probability of joint occurrence between the missing SNP location genotype and tag SNP genotypes in genetically similar individuals.

(C) 3^N *virtual individuals* are generated by using all possible combinations of genotypes of *N* tag SNPs. For each *virtual individual*, we find the *M*-genetically similar individuals from the 1,000 Genomes dataset. We then generate a look-up table, where each index is the sequence of the tag SNP genotypes of a *virtual individual* and the probability of genotypes 0 and 1 for the target SNPs are the columns. These probabilities are calculated using the scoring system described in (B). After converting the look-up table with transformation to the indices and probabilities, we perform *private information retrieval* to the look-up table with encrypted query tag SNP genotypes and return encrypted query target SNP genotype probabilities.

structure using a genotype panel from different populations. Haplotype blocks are then used to determine the relationship between tag and missing SNP genotypes in order to compute a probability for having each genotype (0 for homozygous reference alleles, 1 for heterozygous alternative alleles, and 2 for homozygous alternative alleles) at the missing SNP position. Many of the available imputation methods are based on hidden Markov models in order to determine the size and the boundaries of the haplotype blocks and then impute the genotypes (Howie et al., 2012; Browning and Browning, 2009; Li et al., 2010). Complex statistical methods, such as hidden Markov models, might require computationally expensive operation that may not be implemented using HE-based principles for computation in the encrypted domain. This is because computation may not be able to scale-up genotype imputation for hundreds of individuals due to the large performance overheads.

To avoid some of the overhead problems, we developed a plaintext method that does not depend on phasing or haploblock inference and thus can impute genotypes directly using the tag SNP genotypes. We describe our algorithm in Figure 1B. Let us assume that each haplotype block in the genome contains a fixed number of SNPs. We can further assume that each of the target SNPs to be imputed is conveniently located in the middle of these blocks. That is, for each missing target SNP genotype of a query individual, we can determine N number of tag SNPs that are closest to it. We call this group of SNPs a "pseudo-haplotype block" as they are not necessarily the exact haploblock structures by linkage disequilibrium but an approximation. We can then calculate the joint probability of SNP genotypes on these pseudo-haplotype blocks by using genomes from a reference panel as follows: We define a similarity score between the query individual and each individual k in the known training panel (1,000 Genomes dataset in our case) for the pseudo-haplotype block as:

$$sim(k, query) = \sum_{k=1, j=1}^{k=K, j=N} \left| g_{query}(j) - g_k(j) \right| \qquad \text{(Equation 1)}$$

where K is the total number of individuals in the training dataset, N is the total number of tag SNPs in the pseudo-haplotype block, $g_{query}(j)$ is the genotype of the j^{th} tag SNP in the pseudo-haplotype block of the query individual, and $g_k(j)$ is the genotype of the j^{th} tag SNP in the pseudo-haplotype block of the k^{th} individual in the training dataset. After calculating the similarity between all the individuals and the query individual, we sort the $sim(k, query)$ scores in increasing order and take the top M individuals in the training dataset as our genetically related individuals. From this pseudo-haplotype block, we learn statistical relationships between tag and target SNPs. We then calculate a score for each genotype at the missing SNP location by counting the number of individuals as follows:

$$score(g_{query}(target) = x) = \frac{1}{M} \sum_{j=1}^{N} \sum_{k=1}^{M} I(g_k(target) = x, g_k(j) = g_{query}(j)), \qquad \text{(Equation 2)}$$

where $x = \{0,1,2\}$ and $I(g_k(target) = x, g_k(j) = g_{query}(j))$ are the indicator functions, which equal 1 if k^{th} individual's target SNP genotype is equal to x and j^{th} tag SNP genotype is equal to the query individual's j^{th} tag SNP genotype, and otherwise equals 0 (see STAR Methods for details). We then normalize these scores to obtain the probability of having a genotype at the missing SNP location. The probability of having the genotype x at the missing SNP location becomes

$$p(g_{query}(target) = x) = \frac{score(g_{query}(target) = x)}{\sum_{i=0}^{2} score(g_{query}(target) = i)} \qquad \text{(Equation 3)}$$

Note that the solution above is a simplistic approach compared with the state-of-the-art imputation tools in order to achieve efficiency with a small trade-off in accuracy. This plaintext algorithm cannot be directly used in the encrypted domain due to its reliance of the training database, which requires multiple costly operations. Below, we detail how we further optimized this solution to increase the efficiency in the encrypted domain.

Optimization for an encrypted solution

For this algorithm to work with HE, we must remove the reliance on the training data, as going through all of the individual genomes to infer statistical relationships between SNPs adds a large overhead to the implementation. To this end, we applied a pre-calculated statistical inference as follows: For each pseudo-haplotype block, we enumerate all possible genotype configurations given N tag SNP genotypes and call each a virtual individual. Each tag SNP can have three values: 0, 1, and 2; therefore, we have 3^N possible virtual individuals for each pseudo-haplotype block. We then calculate the genotype probabilities for the target SNP in each pseudo-haplotype block for every virtual individual following the statistical plaintext approach above. We then create a look-up table, where each entry is a virtual individual. The index of the table is a string of concatenated tag SNP genotypes for each virtual individual. The table returns the target genotype probabilities given the string of concatenated tag SNP genotypes for a given pseudo-haplotype block. We create tables for each target SNP separately. These tables can then be encrypted, i.e., given an encrypted tag SNP genotype, it will return the encrypted target genotype probabilities. This is a known problem called "private information retrieval" and can be implemented with $O(n)$ complexity. We describe this algorithm in Figure 1C. The details of the implementation with FHE can be found in the STAR Methods. We used BFV, an FHE scheme, for the following reasons: First, we must use both XNOR and AND gates to be able to perform look-ups in the encrypted domain, which requires both addition and multiplication (see STAR Methods for details). This requirement eliminates the possibility of using PHE. Second, we already know the upper bound of the total number of operations needed, as our tables have fixed sizes due to our parameter N. This allows us to use BFV with known depth and avoid the costly bootstrapping that comes with FHE schemes.

p-Impute is a fast and secure imputation algorithm with robust parameters

The key parameters of the $p - Impute$ algorithm are the number of neighboring tag SNPs (N) and the number of genetically similar individuals (M). Whereas M is only used during building of the look-up table and does not affect the efficiency of our algorithm, N greatly affects the run time and memory requirement of the imputation.

N *is a dataset-independent universal parameter*

As mentioned above, N number of tag SNPs around a target SNP is an approximation to the linkage disequilibrium effect. We aim to capture the tag SNPs that are most highly correlated with the target SNP. This, in principle, will yield similar results when an inference is made for different datasets. To demonstrate this, we carried out a sensitivity analysis by varying the N values and calculated the accuracy of imputation on four datasets: 504 individuals from the 1,000 Genomes Project (The 1000 Genomes Project Consortium, 2015), 836 individuals from the GTEx project (The GTEx Consortium, 2013), 176 individuals from the PsychENCODE project (The PsychENCODE Consortium, 2018), and 1,927 individuals from the ALSPAC project (The UK10K Consortium, 2015). We accomplished this by using

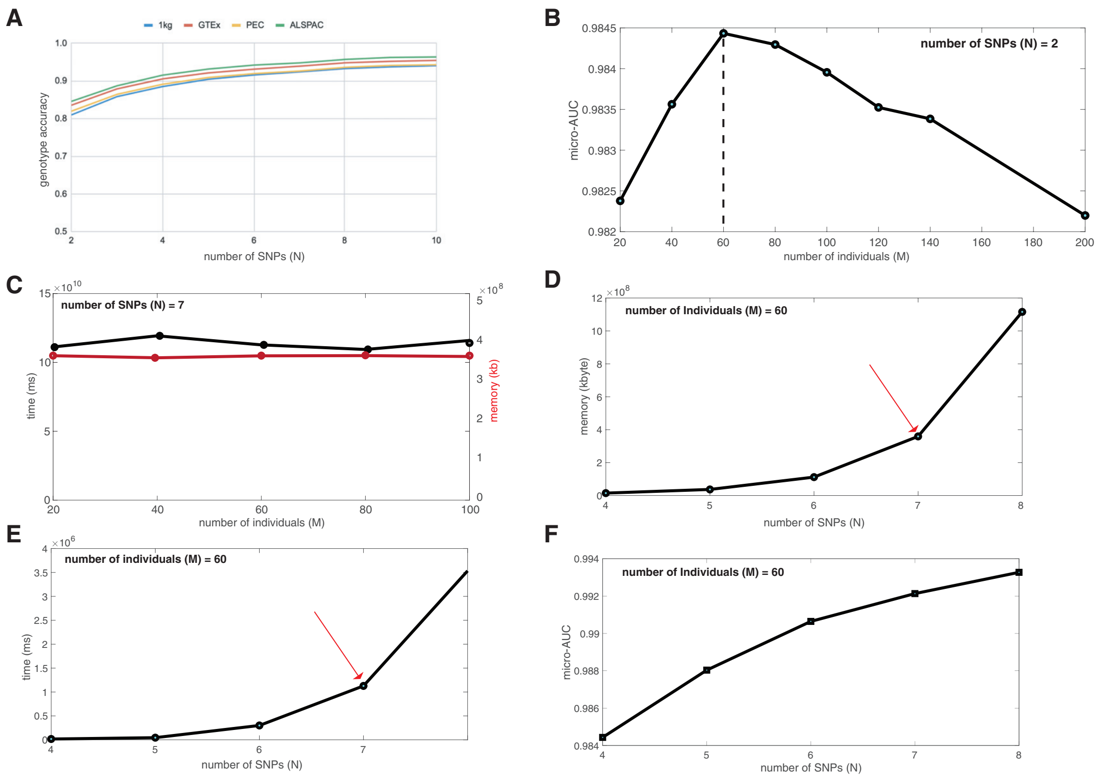

Figure 2. Fine-tuning *N* and M

All of the calculations in this figure were carried out using 9,745 tag and 500 target SNPs of human chromosome 1.

(A) Change in genotype accuracy of the imputation results with different values of *N* for four different datasets. *M* is fixed at 60.

(B) Change in micro-AUC with an increasing number of *M*, with *N* fixed at 2. micro-AUC reaches a maximum at 60 individuals.

(C) Total roundtrip time (encryption+query+decryption) and memory usage with changing M when we fix the number of SNPs at 7. Since M is only used during building the look-up table, as can be seen from the plots, it does not affect the runtime.

(D) Increase in memory usage with increasing numbers of neighboring SNPs when the number of individuals is fixed at 60.

(E) Increase in total roundtrip time (encryption+query+decryption) with an increasing number of neighboring SNPs when the number of individuals is fixed at 60.

(F) Micro-AUC values with an increasing number of *N*, with *M* fixed at 60. micro-AUC increases with an increasing number of neighboring SNPs (N).

~15k tag and ~62k target SNPs on Chr22 (see STAR Methods for details). As shown in Figure 2A, the genotype accuracy of the imputed SNPs behaves the same with an increasing number of SNPs for all four datasets. As expected, the accuracy levels off after a certain number of neighboring tag SNPs, because the correlation between SNPs diminishes as the genomic distance between them increases. This identical behavior of different N values across datasets can primarily be attributed to the fact that "haplotype blocks" containing correlated genotypes are generally consistent across the human population. A haplotype block is defined as a genomic region with no observed recombination; therefore, the structures of these regions are preserved across the human population. Thus, we expect that a predefined *N* number of SNPs used to capture the correlation between genotypes will behave similarly across different human samples. In summary, *N* is a parameter for the human population and does not need to be tuned for a particular dataset.

N *and* M *can be fine-tuned for an optimal balance between efficiency and accuracy*

In order to find an optimum value for these parameters, we performed tests with different values of *N* and *M* and evaluated them in terms of memory and run time. To accomplish this, we divided the 1,000 Genomes dataset into two groups: 2,000 individuals for statistical inference and 504 individuals for parameter tuning using 9,745 tag and 500 target SNPs. For a comprehensive evaluation of the accuracy of the imputation model, we used a metric that reflects both correct predictions (true positive rate) and false misclassifications (false positive rate) for each of the three genotypes. Since this is a prediction problem with three classes, we applied the micro-AUC score, which has been used for comparing the accuracy of various multi-class classification/prediction models (Bradley, 1997). We first fixed *N* at 2 to find the best number of individuals, M. We found that our model had the highest micro-AUC when *M* was 60 individuals (Figure 2B). We then fixed the *M* at 60 and increased *N* from 2 to 8 and measured not only micro-AUC but also the total round trip (encryption+query+decryption) time and memory usage (Figures 2C–2E). When we increased *N* from 7 to 8, we found that micro-AUC increased only by 0.001, whereas the time and memory overhead increased exponentially. For all three performance measures, the optimum value of *N* appears to be 7 in our tests. Since we only used the parameter *M* during statistical inference, we also showed that the run time and memory performance of the query do not depend on *M* (Figure 2C).

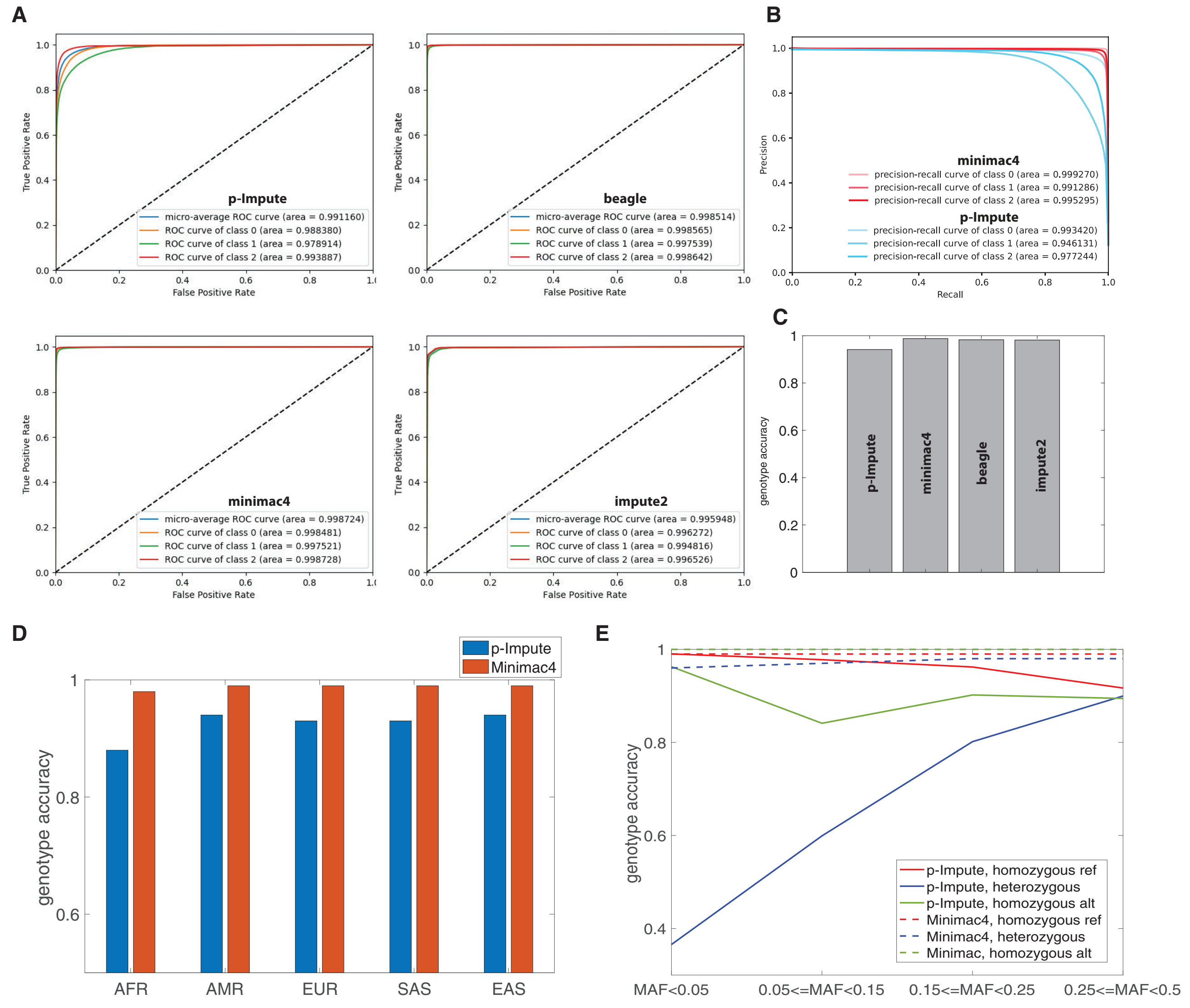

Figure 3. Performance of p-impute ($N = 7; M = 60$), Beagle, IMPUTE2, and Minimac4

(A) The micro-AUC for predicting genotypes of 61,993 target SNPs using 15,424 tag SNPs of Chr22.

(B) Precision-recall curves (shown only against Minimac4 for simplicity) for predicting genotypes of 61,993 target SNPs using 15,424 tag SNPs of Chr22.

(C) Genotype accuracy for predicting genotypes of 61,993 target SNPs using 15,424 tag SNPs of Chr22.

(D)Genotype accuracy for population groups from different ancestries (EUR, Europe; EAS, East Asia; AFR, Africa; AMR, Americas).

(E) Genotype accuracy when the SNPs were grouped by their MAFs, categorized by genotype.

We further compared our prediction with the results obtained from running the commonly used plaintext genotype imputation software IMPUTE2 (Howie et al., 2012), Beagle (Browning and Browning, 2009) and Minimac4 (Das et al., 2016). We trained our method, IMPUTE2, Beagle, and Minimac4 using the genomes of 2,000 individuals from the 1,000 Genomes Project (The 1000 Genomes Project Consortium, 2015) and tested the methods on Chr22 (15,424 tag and 61,993 target SNPs) of 1,927 individuals from the ALSPAC project (The UK10K Consortium, 2015) by plotting the receiver operating curves (ROCs) and precision-recall curves, and then calculating the Micro-AUCs. We found that the performance of $p - Impute$ is in agreement with the performance of the state-of-the-art plaintext methods in terms of micro-AUC (Figure 3A). All four algorithms resulted in a micro-AUC score of 0.99. We found that the $p - Impute$ AUC scores were lowest for the heterozygous SNP genotypes compared with the state-of-the-art methods. For clarity, we compared the area under the precision-recall curves against Minimac4. We found that, albeit still high (>0.95), the precision and recall of $p - Impute$ predictions, especially for heterozygous SNP genotypes, were lower than for Minimac4. We also calculated the genotype accuracy for all four methods (see STAR Methods for details) and found that, albeit still high (0.94), the genotype accuracy of $p - Impute$ was lower than the plaintext state-of-the-art methods (Figure 3C).

We then calculated the genotype accuracy of $p-$ *Impute* on genomes from different ancestries. To ensure enough number of samples per ancestry, we developed our model on 1,500 individuals and tested it on 1,000 individuals from the 1,000 Genomes Project by keeping the ratio of number of samples from different ancestries the same for both the training and test data. We found that genotype accuracy of $p-$ *Impute* for all ancestries except African genomes was above 90%, which is lower than the accuracy values obtained from Minimac4 (99%). Genotype accuracy of $p-$ *Impute* for African genomes was 88%, whereas it was 98% for Minimac4 (Figure 3D). We also calculated the accuracy separately for SNPs with different minor allele frequencies (MAFs). As expected, we found that rare SNP genotypes, especially the heterozygous ones, were generally more difficult to predict. Although $p-$ *Impute* predicted common SNP genotypes with accuracy values comparable with Minimac4, the accuracy for the very rare homozygous SNPs was as low as 37% (Figure 3E).

Lastly, we interrogated the scalability of our tool. Since our implementation allows batching and parallel processing (see STAR Methods for details), we can query multiple target SNPs from multiple individuals at once in the same ciphertext. To demonstrate the benefit of these capabilities on the time and memory requirements, we fixed the number of CPUs at 8 and increased the target SNPs (i.e., the number of missing genotypes in a given genome to be imputed) to measure the time and memory, respectively. Utilizing 870 samples from the GTEx project, when we kept the number of samples to be imputed constant and increased the total number of target SNPs, the time and memory increased linearly (Figure S1). Since we have a fixed amount of data in each ciphertext, the time and memory will always scale linearly with the increasing number of samples as well. Our algorithm is implemented with parallelization. With the large number of CPUs available by cloud providers, achieving extremely fast genome-wide genotype imputation of hundreds of thousands of genomes is possible. That is, as we increase the number of CPUs, the total time and cost of running the algorithm will decrease linearly.

Limitations of *p-Impute*

The efficiency and privacy-preservation nature of the algorithm come with a cost on accuracy. As can be seen in Figure 3, our algorithm cannot impute heterozygous genotypes (especially the rare ones), as well as the state-of-the-art plaintext methods. This can be attributed to the lack of a phasing step, which makes it possible to determine to which of the haplotypes the alternative allele belongs. In traditional plaintext genotype imputation algorithms, the correlation between the presence or absence of the alternative alleles of the tag and target SNP is determined after the phasing step. For example, let us assume we have a tag and a target SNP, both of which have the heterozygous genotype. Without knowing if the alternative alleles of these SNPs are on maternal or paternal haplotypes, we cannot conclude that the presence of these SNP alleles is correlated. It could be that we never observe both of the alternative alleles on the same haplotype, meaning the presence of the tag SNP alternative allele is correlated with the absence of the target SNP alternative allele. Therefore, the lack of a phasing step results in lower accuracy for heterozygous SNPs while making the algorithm faster.

DISCUSSION

Privacy of an individual's genomic data has emerged as a major focus of data privacy studies, as unrestricted availability of personal genetic information gives rise to many concerns. For example, knowledge of genetic predisposition to diseases may bias insurance companies or create unlawful discrimination by employers. In addition to DNA sequencing data, recent studies have shown that high-throughput molecular phenotype datasets, such as functional genomic, metabolomic, or even microbiome measurements, can increase the number of quasi-identifiers and the possibility of re-identification by an adversary. The rise in popularity of genetic and ancestry testing companies, which collect and distribute large amounts of genomics and health data, will only increase privacy concerns over sharing personal biological data.

In contrast, human genetics studies mainly focus on identifying genetic variants that affect human traits and diseases. This identification is possible by collecting and analyzing genetics data from large cohorts of individuals with different phenotypes. In particular, genome-wide association studies and studies focused on identifying various quantitative trait loci (e.g., GTEx and PsychENCODE) aim to genotype thousands of individuals to better characterize human diseases. There are two cost-associated bottlenecks for large-scale genomics studies. The first is regarding the feasibility of comprehensively genotyping entire genomes. Although the cost of genome sequencing is rapidly decreasing, it is still not feasible to perform high-coverage whole-genome sequencing on thousands of genomes. To overcome this, researchers have developed genotype imputation tools that can predict the missing genotypes using available population genetics data. Although genotype imputation is an extremely powerful technique, it leads to a second bottleneck of computationally storing and imputing thousands of genomes on local servers and computers. Therefore, institutions and funding agencies are increasingly outsourcing cloud services for complex genomics analyses, such as genotype imputation.

Outsourcing to third-party computing environments for storage and analysis of individuals' genetic data raise serious privacy concerns. There is an increasing need for fast and scalable privacy-preserving genomic analysis tools that keep the genetic data encrypted in third-party computing environments. This can be achieved by a special type of encryption, called HE. HE allows for the manipulation of encrypted data, hence removing the privacy risk associated with the decryption of sensitive data for computation purposes. However, there is a large computational overhead incurred in HE calculations compared with computations on plaintext. This cost depends on the class of computation needed on the encrypted variables. When arbitrary computation is required, which is the case for most genomic calculations, FHE must be employed. However, this leads to tremendous overhead, slowing the process by several orders of magnitude. Therefore, it is important to design fast and scalable algorithms that can be implemented using HE.

In real-world applications, these scenarios can be observed in both research and clinical settings. For example, the NIH/NHGRI has invested a tremendous amount of resources in the AnVIL infrastructure. This effort aims to provide a framework to analyze large, open, and controlled-access genomic datasets with familiar tools and reproducible workflows in a cloud-based computing environment. Projects funded by the NHGRI are expected to eventually move to AnVIL. However, this creates serious privacy issues when we carry out computations on genomes that we cannot share on cloud services due to lack of consent. Even in situations in which we obtain consent, different regional laws, such as GDPR (General Data Protection Regulation) or CCPA (California Consumer Privacy Act), might only allow sharing personal genetic data under encryption (Scheibner et al., 2021). Thus, we propose that privacy-preserving genotype imputation will overcome these concerns and allow researchers to impute genotypes from private genomes using cloud infrastructures. Similarly, private data from patients can also be encrypted and imputed in cloud settings without the need to locally download and run complicated models. For example, let us assume a new consortium aims to sequence 10,000 genomes; however, due to cost limitations, they decide to use gene chip technology to genotype a subset of variants with the hope that they can impute the rest of the genome. In the meantime, it becomes challenging to find 10,000 individuals who will give broad consent to the use of their genetic data. Since computing genotype imputation on 10,000 individuals will create logistical issues with the movement and storage of highly sensitive and private data, they can outsource the storage and the computation to the cloud provider by encrypting the data and performing the genotype imputation on the encrypted genomes. As this is a fairly large compute, with our highly parallelizable private genotype imputation model, one can take a cloud burst approach to spin up many CPUs to perform genotype imputation on a large number of samples in an efficient and cost-effective way.

In this study, we developed a scalable, privacy-preserving genotype imputation method using the BFV encryption scheme. One of the key parameters in our algorithm is the selection of the number of the neighboring SNPs (N) to harbor the statistics for the genotypes. Since N greatly affects the efficiency of the algorithm, it is important to find an optimum value. When we tested different N values across different datasets, we observed a similar behavior in accuracy. We think this can mainly be attributed to the fact that "haplotype blocks" that contain correlated genotypes are consistent across the human population. A haplotype block is defined as a genomic region with no observed recombination; therefore, these regions are conserved across the human population. Thus, it is expected that a predefined "N" number of SNPs used to capture the correlation between genotypes will behave similarly across different human samples. However, as we collect more samples from diverse ancestry groups, we must retrain these imputation algorithms as the sizes of the haplotype blocks and the correlation between different SNPs may no longer hold.

We showed that our imputation results are comparable with results from plaintext methods when tested on independent datasets. We showed that we can achieve chromosome-wide imputation for 1,000 individuals in under 1.5 days (30 h in using 8 cores in a Linux workstation with Intel Xeon E5-2680 CPU at 2.4 GHz) in the encrypted domain for up to 1,000 genomes. The client spends 3 s for key generation, 3 s for encryption. The server spends 1.3 days on the computation. The client then spends another 8 s for the decryption. Note that although plaintext genotype imputation methods require an additional phasing step, their performance in terms of timing and memory is still lower than $p-$ *Impute*. All three plaintext methods tested in this study (i.e., Minimac4, Beagle, and IMPUTE2) achieved the same chromosome-wide imputation in under 20 h, which has been previously benchmarked (Shi et al., 2018).

We used the BFV scheme (Fan and Vercauteren, 2012) in our FHE implementation to leverage its batching capability. With batching, we were able to fit many plaintexts into a single ciphertext, enabling processing of thousands of ciphertexts per operation. Although schemes like torus FHE (TFHE) (Chillotti et al., 2020) are faster when processing 1 bit of information, the batching capability of BFV allows us to process many bits of information more efficiently. Other schemes, such as Brakerski-Gentry-Vaikuntanathan (BGV) (Brakerski et al., 2012), also allow batching.

In this study, we addressed the issue of protecting the sensitive input and the output of genotype imputation in a cloud computing setting by using HE as the privacy mechanism. Our inference is based on the statistics we derived locally and from broadly shared, publicly available 1,000 Genomes data. However, in the case of building inference models from private genomes (e.g., from the TOPmed dataset, Taliun et al., 2021) one also has to consider model privacy. For instance, studies have shown that private information in training data can be gleaned from models via membership inference attacks (Shokri et al., 2017). In such cases, other privacy mechanisms, such as differential privacy in a federated learning system (Wei et al., 2020), might be more suitable.

There are many different ways to provide privacy preservation to algorithms depending on the privacy requirement. We chose to use FHE for the genotype imputation problem because it provides privacy preservation to both the input and output of the algorithm (i.e., the SNPs of an individual). Other techniques that enable privacy preservation include secure multiparty computation (SMC) (Yao, 1986), functional encryption (Boneh et al., 2011), trusted executive engines (TEEs) (Sabt et al., 2015), and federated learning (Bonawitz et al., 2019). SMC aims to protect the inputs of the parties, whereas the output is learned by all parties involved. Moreover, it is more suitable for problems that require collaborative computing on data distributed across sites (Cho et al., 2018; Hie et al., 2018). Similarly, functional encryption protects the input only and provides the output in plaintext; this method is also slower than FHE. Federated learning is useful for machine learning applications with decentralized data, such as those of mobile devices. In federated learning, the parties train a machine learning model locally using their private data and send it back to update the overall model, which is re-shared with the parties. This is an iterative process that refines the model. Such an approach does not apply to our problem due to the limited number of parties (Alice has all the private data) and the fact that Bob may want to keep his model/weights private and Alice may not want to perform computation locally.

With the advent of computationally efficient algorithms and the widespread use of cloud computing, there has been an increasing interest in privacy-preserving genotype imputation. For example, TEEs have recently been a popular privacy-preserving approach for computing on sensitive genomes (Kockan et al., 2020; Hie et al., 2018). A recent implementation of genotype imputation with a TEE results in high accuracy (Dokmai et al., 2021). However, this implementation requires the genotypes to be phased locally before performing the imputation in the enclave, which hampers its applicability and usability. Moreover, using TEEs for private computation requires the client to trust the hardware provider, such as Intel, with the sensitive data, making their security levels less desirable than homomorphic encryption. Another recent study (Kim et al., 2021), similar to ours, offered multiple homomorphic encryption-based solutions to privacy-preserving genotype imputation. These solutions have many advantages, such as extremely high-speed computation and low memory requirements. Although the overall accuracy of the presented solutions seems to be comparable with our solution, this study lacks extensive benchmarking. Specifically, (1) training (1,500 individuals) and testing (1,000 individuals) are performed on uniformly collected and processed 1,000 Genomes samples, which are limited in size compared with the datasets used in this study (total of 3,443 samples spanning four different studies: 1,000 Genomes, GTEx, psychENCODE, and ALSPAC). It is important to benchmark the algorithms against many different datasets in order to clarify whether the parameters tuned in the machine-learning training step will have a dataset-specific bias. (2) We also suspect the presented methods will also suffer from a reduction in accuracy of imputing rare non-reference alleles due to lack of phasing and the nature of machine learning capturing common patterns, which was not presented. We believe this is also an important benchmark to perform as it can help clients to choose a subset of variants to be imputed locally with more accurate imputation techniques, while they choose to impute other larger subset in the cloud with privacy preservation.

STAR★METHODS

Detailed methods are provided in the online version of this paper and include the following:

- KEY RESOURCES TABLE
- RESOURCE AVAILABILITY
  - Lead contact
  - Materials availability
  - Data and code availability
- METHOD DETAILS
  - Homomorphic encryption
  - Brakerski Fan Vercauteren (BFV) encryption system
  - Security
  - *p-Impute* details
  - Performance evaluation metrics
  - Dataset

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.cels.2021.10.003.

ACKNOWLEDGMENTS

We thank Center of Cyber Security at New York University Abu Dhabi. This study was supported by grants from the NIH (R01 HG010749 to M.G.; K99 HG010909 to G.G.). This work is also supported by the AL Williams Professorship fund.

AUTHOR CONTRIBUTIONS

G.G., E.C., M.M., and M.G. conceived of the study. G.G. and E.C.. designed and implemented the p-Impute software. G.G. and C.M.B. analyzed the results. G.G., E.C., C.M.B., M.M., and M.G. wrote the manuscript.

DECLARATION OF INTERESTS

The authors declare no competing interests.

Received: August 21, 2020
Revised: June 28, 2021
Accepted: October 15, 2021
Published: November 9, 2021

REFERENCES

The 1000 Genomes Project Consortium (2015). A global reference for human genetic variation. Nature *526*, 68–74.

Acar, A., Aksu, H., Uluagac, A.S., and Conti, M. (2018). A survey on homomorphic encryption schemes: theory and implementation. ACM Comput. Surv. *51*, 1–35. https://doi.org/10.1145/3214303.

Albrecht, M., Chase, M., Chen, H., Ding, J., Goldwasser, S., Gorbunov, S., Halevi, S., Hoffstein, J., Laine, K., Lauter, K., et al. (2018). Homomorphic encryption security standard. Technical report. https://homomorphicencryption.org/.

Armknecht, F., Boyd, C., Carr, C., Gjøsteen, K., Jaschke, A., Reuter, C.A., and Strand, M. (2015). A guide to fully homomorphic encryption. Cryptology eprint Archive, Report 2015/1192. https://eprint.iacr.org/2015/1192.

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konecny, J., Mazzocchi, S., McMahan, H.B., et al. (2019). Towards federated learning at scale: system design. arXiv, arXiv:1902.01046.

Boneh, Dan, Sahai, Amit, and Waters, Brent (2011). Functional Encryption: Definitions and Challenges. Theory of Cryptography Conference, https://doi.org/10.1007/978-3-642-19571-6_16.

Bos, J.W., Lauter, K., and Naehrig, M. (2014). Private predictive analysis on encrypted medical data. J. Biomed. Inform. *50*, 234–243.

Bradley, A.P. (1997). The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern Recognit *30*, 1145–1159.

Brakerski, Z., Vaikuntanathan, V., and Gentry, C. (2012). Fully homomorphic encryption without bootstrapping. In ITCS '12: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, pp. 309–325.

Browning, B.L., and Browning, S.R. (2009). A unified approach to genotype imputation and haplotype-phase inference for large data sets of trios and unrelated individuals. Am. J. Hum. Genet. *84*, 210–223.

Cheon, J.H., Kim, A., Kim, M., and Song, Y. (2016). Homomorphic encryption for arithmetic of approximate numbers. In Advances in Cryptology – ASIACRYPT 2017 (Springer), pp. 409–437.

Cheon, J.H., Coron, J.-S., Kim, J., Lee, M.S., Lepoint, T., Tibouchi, M., and Yun, A. (2013). Batch fully homomorphic encryption over the integers. In Advances in Cryptology – EUROCRYPT 2013, T. Johansson and P.Q. Nguyen, eds. (Springer), pp. 315–335.

Chillotti, I., Gama, N., Georgieva, M., and Izabachène, M. (2020). Tfhe: fast fully homomorphic encryption over the torus. J. Cryptol. *33*, 34–91.

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

Chor, B., Kushilevitz, E., Goldreich, O., and Sudan, M. (1998). Private information retrieval. J. ACM *45*, 965–981. https://doi.org/10.1145/293347.293350.

Das, S., Forer, L., Schönherr, S., Sidore, C., Locke, A.E., Kwong, A., Vrieze, S.I., Chew, E.Y., Levy, S., McGue, M., et al. (2016). Next-generation genotype imputation service and methods. Nat. Genet. *48*, 1284–1287.

Dokmai, N., Kockan, C., Zhu, K., Wang, X., Sahinalp, S.C., and Cho, H. (2021). Privacy-preserving genotype imputation in a trusted execution environment.

Cell Syst *12*, 983–993.e7. https://www.sciencedirect.com/science/article/pii/S2405471221002891.

Fan, J., and Vercauteren, F. (2012a). Somewhat practical fully homomorphic encryption. Cryptology eprint Archive, Report 2012/144. https://eprint.iacr.org/2012/144.pdf.

Gentry, C. (2010). Computing arbitrary functions of encrypted data. Commun. ACM *53*, 97–105.

GTEx Consortium (2013). The genotype-tissue expression (gtex) project. Nat. Genet. *45*, 580–585.

Halevi, S., and Shoup, V. (2015). Bootstrapping for helib. In Advances in cryptology – EUROCRYPT 2015′, M.F. Elisabeth Oswald, ed. (Springer), pp. 641–670.

Hie, B., Cho, H., and Berger, B. (2018). Realizing private and practical pharmacological collaboration. Science *362*, 347–350.

Howie, B., Fuchsberger, C., Stephens, M., Marchini, J., and Abecasis, G.R. (2012). Fast and accurate genotype imputation in genome-wide association studies through pre-phasing. Nat. Genet. *44*, 955–959.

Kim, L., and Lauter, K. (2015). Private genome analysis through homomorphic encryption. BMC Med. Inform. Decis. Mak. *15*, S3.

Kim, M., Harmanci, A.O., Bossuat, J.P., Carpov, S., Cheon, J.H., Chillotti, I., Cho, W., Froelicher, D., Gama, N., Georgieva, M., et al. (2021). Ultrafast homomorphic encryption models enable secure outsourcing of genotype imputation. Cell Syst. https://www.sciencedirect.com/science/article/pii/S240547122100288X.

Kocabaş, Ö., and Soyata, T. (2014). Medical data analytics in the cloud using homomorphic encryption. In E-Health and Telemedicine: Concepts, Methodologies, Tools, and Applications (IGI Global), pp. 751–768.

Kockan, C., Zhu, K., Dokmai, N., Karpov, N., Kulekci, M.O., Woodruff, D.P., and Sahinalp, S.C. (2020). Sketching algorithms for genomic data analysis and querying in a secure enclave. Nat. Methods *17*, 295–301.

Li, Y., Willer, C.J., Ding, J., Scheet, P., and Abecasis, G.R. (2010). Mach: using sequence and genotype data to estimate haplotypes and unobserved genotypes. Genet. Epidemiol. *34*, 816–834.

Lyubashevsky, V., Peikert, C., and Regev, O. (2012). On ideal lattices and learning with errors over rings. Cryptology eprint Archive, Report 2012/230. https://eprint.iacr.org/2012/230.

Manolio, T.A. (2010). Genomewide association studies and assessment of the risk of disease. N. Engl. J. Med. *363*, 166–176.

McLaren, P.J., Raisaro, J.L., Aouri, M., Rotger, M., Ayday, E., Bartha, I., Delgado, M.B., Vallet, Y., Günthard, H.F., Cavassini, M., et al. (2016). Privacy-preserving genomic testing in the clinic: a model using hiv treatment. Genet. Med. *18*, 814–822.

Micciancio, D. (1997). Oblivious data structures: applications to cryptography. In STOC '97: proceedings of the twenty-ninth annual ACM symposium on theory of computing, pp. 456–464.

National Human Genome Research Institute (2021). The NHGRI genomic data science analysis, visualization and informatics lab-space (AnVIL). https://www.genome.gov/Funded-Programs-Projects/Computational-Genomics-and-Data-Science-Program/Genomic-Analysis-Visualization-Informatics-Lab-space-AnVIL.

Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. In Advances in cryptology — EUROCRYPT'99, J. Stern, ed. (Springer), pp. 223–238.

PsychENCODE Consortium (2018). Revealing the brain's molecular architecture. Science *362*, 1262–1263.

Rockman, M.V., and Kruglyak, L. (2006). Genetics of global gene expression. Nat. Rev. Genet. *7*, 862–872.

Sabt, M., Achemlal, M., and Bouabdallah, A. (2015). Trusted execution environment: what it is, and what it is not. In 2015 IEEE Trustcom/BigDataSE/ISPA, pp. 57–64.

Sboner, A., Mu, X.J., Greenbaum, D., Auerbach, R.K., and Gerstein, M.B. (2011). The real cost of sequencing: higher than you think. Genome Biol *12*, 125.

Scheibner, J., Raisaro, J.L., Troncoso-Pastoriza, J.R., Ienca, M., Fellay, J., Vayena, E., and Hubaux, J.P. (2021). Revolutionizing medical data sharing using advanced privacy-enhancing technologies: technical, legal, and ethical synthesis. J. Med. Internet Res. *23*, e25120. https://www.jmir.org/2021/2/e25120.

SEAL (2019). Microsoft SEAL (release 3.3.2) (Microsoft Press Research). https://github.com/Microsoft/SEAL.

Shi, S., Yuan, N., Yang, M., Du, Z., Wang, J., Sheng, X., Wu, J., and Xiao, J. (2018). Comprehensive assessment of genotype imputation performance. Hum. Hered. *83*, 107–116.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. (2017). Membership inference attacks against machine learning models. In 2017 IEEE Symposium on Security and Privacy (SP), pp. 3–18.

Taliun, D., Harris, D.N., Kessler, M.D., Carlson, J., Szpiech, Z.A., Torres, R., Taliun, S.A.G., Corvelo, A., Gogarten, S.M., Kang, H.M., et al. (2021). Sequencing of 53,831 diverse genomes from the nhlbi topmed program. Nature *590*, 290–299.

The UK10K Consortium (2015). The UK10K project identifies rare variants in health and disease. Nature *526*, 82–90.

Wei, K., Li, J., Ding, M., Ma, C., Yang, H.H., Farokhi, F., Jin, S., Quek, T.Q.S., and Poor, H.V. (2020). Federated learning with differential privacy: algorithms and performance analysis. IEEE Trans.Inform. Forensic Secur. *15*, 3454–3469.

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

STAR★METHODS

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Software and algorithms		
p-Impute software	This paper	https://doi.org/10.5281/zenodo.5542001
Python (micro-AUC curves)	The Python software foundation	https://www.python.org
C++ programming language	ISO/IEC JTC1 (Joint Technical Committee 1) / SC22 (Subcommittee 22) / WG21 (Working Group 21)	http://www.open-std.org/jtc1/sc22/wg21/
SEAL Library	Microsoft	https://www.microsoft.com/en-us/research/project/microsoft-seal/
Data		
The 1000 Genomes Project WGS data	The 1000 Genomes Consortium	https://www.internationalgenome.org/data
GTEx Project WGS data	The Genotype-Tissue Expression Consortium Portal	**dbGaP Study Accession**: phs000424.v8.p2
psychENCODE WGS data	The psychENCODE Consortium Knowledge Portal	https://doi.org/10.7303/syn4921369
UK10K ALSPAC cohort WGS data	The UK10K Project at European Genome-Phenome Archive	**EGA Dataset ID**: EGAD00001000789
Illumina Duo 1M version 3 SNP-chip	Illumina Inc.	https://support.illumina.com/downloads/human1m-duo_v3-0_product_files.html

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources and reagents should be directed to and will be fulfilled by the lead contact, Mark Gerstein (mark@gersteinlab.org).

Materials availability

This study did not generate new materials.

Data and code availability

- This paper analyzes existing, publicly available data. These accession numbers for the datasets are listed in the key resources table.
- All original code has been deposited at https://github.com/gersteinlab/idash19he and is publicly available as of the date of publication. DOIs are listed in the key resources table.
- Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon request.

METHOD DETAILS

Homomorphic encryption

HE makes it possible to compute the encrypted data directly without decryption. There are a variety of mathematical models for HE, each of which supports a different kind and number of operations. Some of these models support one operation such as the Paillier encryption scheme (Paillier, 1999) of PHE. By contrast, the FHE scheme supports more than one operation to theoretically allow all possible computations of recursive functions.

The security of the most common FHE schemes is provided by the hardness of the Ring Learning With Errors (RLWE) problem (Lyubashevsky et al., 2012). The hardness of the RLWE problem is due to adding a small amount of error to a point in a lattice, which makes it difficult to determine to which point the error was added. This creates noise; hence, the ciphertexts in HE schemes are noisy, which grows during homomorphic additions and multiplications. This growth eventually makes it impossible to decrypt the resulting ciphertext in FHE. To overcome this, Gentry proposed a bootstrapping technique that can evaluate its own decryption function (Gentry, 2010). Bootstrapping re-creates a ciphertext by running the decryption function on it homomorphically with an encrypted secret key, which reduces the noise.

Although bootstrapping is extremely helpful in deep arithmetic circuits, if the number of operations needed in an algorithm is known and small, bootstrapping may not be needed. Some of the leveled FHE constructions do not use bootstrapping procedures. A leveled

FHE scheme can evaluate L-level arithmetic circuits with $O(\lambda \cdot L^3)$ per-gate computation. Security is also based on RLWE and the parameters for the desired security level are derived by taking L into account (Fan and Vercauteren, 2012).

Data in these encryption schemes are represented by polynomials both when they are encrypted (the ciphertext) and when they are unencrypted (plaintext). A special polynomial called the polynomial modulus is defined, and only the remainder of polynomials is considered when they have been divided by this polynomial modulus.

Brakerski Fan Vercauteren (BFV) encryption system

The BFV encryption system (Fan and Vercauteren, 2012) is a lattice-based cryptographic scheme that depends on the hardness of the RLWE problem. With BFV, one can perform both addition/subtraction and multiplication within the encrypted domain. Divisions and exponentiation of a number by an encrypted one and non-polynomial operations are not supported. The computations can only be performed on integers. Equation 4 depicts the concept of homomorphism, where c_1 and c_2 are ciphertexts, E and D are encryption and decryption functions, respectively, and $\otimes$ and $\bigstar$ are homomorphically equivalent operators, i.e., $\otimes$ over ciphertexts is equivalent to the encryption of $\bigstar$ over plaintexts.

$$c_1 \otimes c_2 = E(D(c_1) \bigstar D(c_2)) \qquad \text{(Equation 4)}$$

Bootstrapping is supported by BFV, but due to its inefficiency, it is not commonly implemented by FHE libraries such as Microsoft SEAL, making it in practice an SHE scheme. This limitation leads to the need for knowing the depth of the arithmetic circuit for properly selecting the encryption parameters.

The specific form of this polynomial modulus in the BFV scheme is $x^d + 1$. where $d = 2^n$ for some n.

In this study, we used BFV as our FHE scheme. There are other schemes that can be used, such as The Fast Fully Homomorphic Encryption over the Torus (TFHE) (Chillotti et al., 2020), the Brakerski-Gentry-Vaikuntanathan (BGV) (Brakerski et al., 2012), and Cheon-Kim-Kim-Song (CKKS) (Cheon et al., 2016). TFHE is a faster library when processing 1 bit of information, since it operates on gates. However, when processing can be parallelized, BFV has an advantage because it supports batching (i.e., many plaintexts fit into one ciphertext). When considering batching, BFV can process thousands of plaintexts per operation. TFHE does not support batching. BGV also supports batching,

In BFV, the plaintext modulus is large (e.g., $2^{16} + 1$), which enables higher precision without affecting performance. CKKS implements fixed-point arithmetic, therefore, it is not ideal for the bit-level arithmetic required by our algorithm.

Security

The security of our implementation depends only on the security of the BFV encryption scheme, which is based on the RLWE problem (Lyubashevsky et al., 2012). We selected encryption parameters that provide enough noise budget for the computation and confer 128 bits of security, which is considered secure by the Homomorphic Encryption Security Standard (Albrecht et al., 2018). The plaintext modulus is set to 65,537 in order to enable batching. The degree of the polynomial modulus n and the coefficient modulus size $\log_2(q)$ varies according to the number of select tag SNPs N. For $N <= 4$, we use $n = 2^{14}$ and $\log_2(q) = 438$ bits, while for $N > 4$, we use $n = 2^{15}$ and $\log_2(q) = 881$ bits.

p-Impute details

Encryption

For each target SNP, we select from each query individual only the tag SNPs relevant for that target SNP (i.e., the N closest ones). Let T be the number of target SNPs. Then, the total number of tag SNPs per individual to be encrypted is given by $N \cdot T$, which is much less than the number of tag SNPs available. Furthermore, each tag SNP contains a value in the set {0, 1, 2}, which we break into its bit representation {00, 01, 10} to allow encrypted comparison during querying. Thus, we have $2NT$ plaintexts per query individual.

Our solution requires addition, multiplication, and comparison on encrypted data. We used the BFV encryption scheme (Fan and Vercauteren, 2012). The BFV encryption scheme supports homomorphic addition, subtraction, and multiplication, while comparison can be implemented using homomorphic logic gates, which, in turn, can be implemented using addition, subtraction, and multiplication. The implementation of BFV available in SEAL does not support bootstrapping. However, it is not required since we know a priori the number of homomorphic operations executed by our solution. The BFV scheme also supports batching, which allows us to combine many plaintexts into one ciphertext and operate on all plaintexts at the same time, in a single-instruction multiple-data fashion. The number of slots available in a ciphertext is given by the degree of its polynomial modulus, a number in the thousands. We maximize the usage of those slots by packing multiple query individuals into a ciphertext. We pack only independent information in the same ciphertext, i.e., each ciphertext contains only one bit of information related to each target SNP per individual. For example, if we have 500 target SNPs and 2^{14} slots, we can put 32 individuals into the same ciphertext. Because each target SNP requires N tag SNPs and the tag SNPs are broken into two bits each, we need $2N$ ciphertexts to represent those 32 individuals. The number of individuals within a ciphertext is given by $\frac{slots}{T}$.

Query

Since the result of our statistical inference is a set of look-up tables, our querying consist of indexing these tables. Searching on a table where the query is encrypted is known as the *Private Information Retrieval* problem (Chor et al., 1998). In the encrypted domain, the complexity of the search (vanilla implementation) is linear to the number of elements in the table ($O(n)$), while exponential in the number of tag SNPs (3^N is the number of elements in the look-up table, N is the number of tag SNPs selected for each target SNP).

Despite having one lookup table per target SNP, we can search on all of them in parallel due to batching. Batching is a technique that supports encrypting and homomorphically processing a vector of plaintext bits as a single ciphertext (Cheon et al., 2013), as described above in the encryption section. Furthermore, all target SNPs of several query individuals are searched in parallel, since they are packed in the same ciphertext in accordance with the encryption methodology.

The first step is to prepare the look-up table for querying. We scale the probabilities by a multiplying factor, convert them to integers, and then pack them together for batching. We represent both probabilities (of genotypes 0 and 1) in a single plaintext. We do that by using *shift left* and *add* (e.g., $P_{1,0} = (P(1) \ll s) + P(0)$, where s is the shifting constant). Thus, we get all probabilities in a single search, improving performance and reducing memory usage. In our implementation, we use $s = 7$.

In order to perform comparisons in the encrypted domain, we must convert the data to a binary representation, so we can emulate gate operations on top of homomorphic addition, subtraction, and multiplication. This is the reason that we break the genotypes of the tag SNPs into their binary constituents (i.e., 0 becomes 00, 1 becomes 01, and 2 becomes 10). We can implement equality using the $2N$ XNOR gates ($xXNORy = 1 + 2xy - (x + y)$) between the query individual and the table index, and then apply $2N - 1$ AND gates ($xANDy = xy$) to get a binary output. The result of the equality is multiplied by the combined probabilities. If the equality results in the encryption of 1, the result of the multiplication is the probabilities. Otherwise, it is the encryption of zero. Each comparison of the query to a table index will either result in the probabilities for that index or zero. Since it is a look-up table of all possible selected tag SNPs, there is always one and only one match. Therefore, the addition of all partial results is the value of the indexed query. In addition, since AND gate and multiplication are the same operation, we embedded the multiplication between the result of the equality and the value of the table index inside the equality to reduce the depth of the circuit, as deeper circuits produce more noise. In addition, we add thread-level parallelism to the query, where we simply divide the workload (searches) among different threads.

Decryption

To obtain the probabilities and resulting target SNP genotypes, we decrypt the ciphertexts, unpack the data, decouple the probabilities (*shift right* and *mask*), convert them back to floating-point, and scale back using the multiplying factor we used for encryption. Then, we calculate the probability for genotype 2 ($P(2) = 1 - P(1) - P(0)$). The target SNP genotype is the genotype with the highest probability.

Performance evaluation metrics

Micro-AUC

Micro-AUC is used for multi-class prediction tasks. Because we have three classes of genotypes (i.e., 0, 1, and 2), micro-AUC computes the AUC for each class and then aggregates all three AUCs for computing the overall AUC of the prediction. This was also used at the iDASH19 competition (Kim et al., 2021) as the metric for performance evaluation. We used a similar approach when we calculated the precision-recall curves.

Genotype accuracy

After assigning the highest probability genotype to a SNP, the genotype accuracy is calculated as the ratio between the total number of correctly predicted genotypes and the total number of target SNPs.

Variant accuracy

Variant accuracy calculates the ratio between the total number of individuals for which the genotype is correctly predicted per variant and the total number of individuals per variant.

Dataset

Following ref. (Kim et al., 2021), we have obtained our tag SNPs using the SNPs assayed in Illumina Duo 1M version 3 SNP-chip (https://support.illumina.com/downloads/human1m-duo_v3-0_product_files.html). This allowed us to test our model in a realistic setting. With this assay, we have obtained ~15k tag SNPs and imputed ~ 62k target SNPs on Chr 22. We used fully characterized genomes from 2,000 individuals provided by the 1000 Genomes Project (The 1000 Genomes Project Consortium, 2015) as our complete dataset to infer statistical relationships between SNP genotypes. We then evaluated our predictions using 500 individuals from the 1000 Genomes Project (The 1000 Genomes Project Consortium, 2015), 836 individuals from the GTEx project (The GTEx Consortium, 2013), 176 individuals from the PsychENCODE project (The PsychENCODE Consortium, 2018), and 1,927 individuals from the ALSPAC project (The UK10K Consortium, 2015).