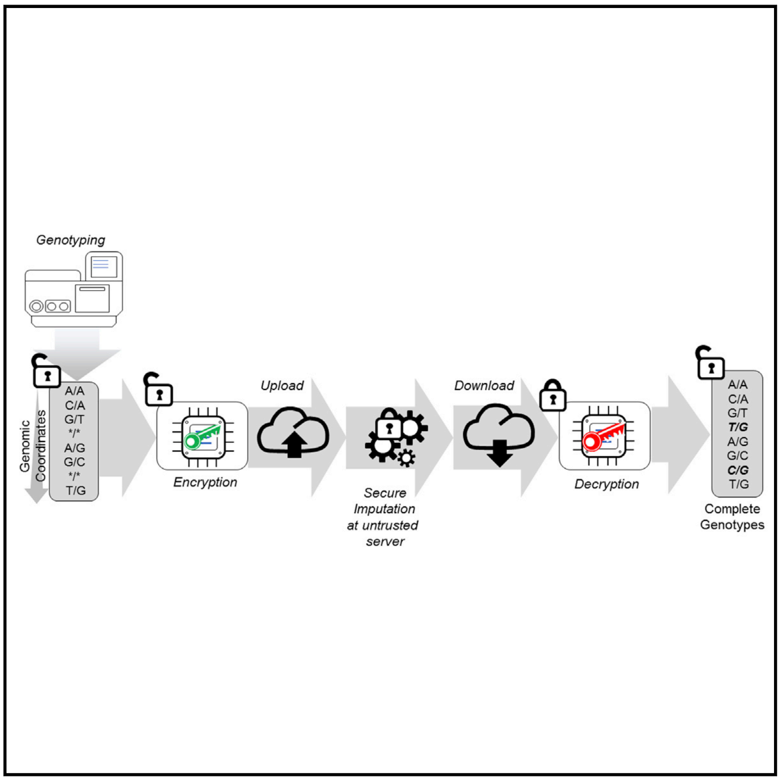
Ultrafast homomorphic encryption models enable secure outsourcing of genotype imputation

Graphical abstract



Authors

Miran Kim, Arif Ozgun Harmanci, Jean-Philippe Bossuat, ..., Yongsoo Song, Juan Troncoso-Pastoriza, Xiaoqian Jiang

Correspondence

arif.o.harmanci@uth.tmc.edu (A.O.H.), xiaoqian.jiang@uth.tmc.edu (X.J.)

In brief

Kim et al. present fast and efficient methods for privacy-aware outsourcing of genotype imputation. The presented secure analysis frameworks can be adopted by other high-throughput genomic data analysis tools.

Highlights

- Fast homomorphic encryption enables secure and practical genotype imputations
- Secure methods require comparable resources as non-secure methods
- Accuracy of secure methods can be improved using population specific panels

Cell Systems

CellPress
OPEN ACCESS

Methods

Ultrafast homomorphic encryption models enable secure outsourcing of genotype imputation

Miran Kim,[1,16] Arif Ozgun Harmanci,[2,16,17,*] Jean-Philippe Bossuat,[3] Sergiu Carpov,[4,5] Jung Hee Cheon,[6,7] Ilaria Chillotti,[8] Wonhee Cho,[6] David Froelicher,[3] Nicolas Gama,[4] Mariya Georgieva,[4] Seungwan Hong,[6] Jean-Pierre Hubaux,[3] Duhyeong Kim,[6] Kristin Lauter,[9] Yiping Ma,[10] Lucila Ohno-Machado,[11] Heidi Sofia,[12] Yongha Son,[13] Yongsoo Song,[14] Juan Troncoso-Pastoriza,[3] and Xiaoqian Jiang[15,*]

[1]Department of Computer Science and Engineering and Graduate School of Artificial Intelligence, Ulsan National Institute of Science and Technology, Ulsan 44919, Republic of Korea
[2]Center for Precision Health, School of Biomedical Informatics, University of Texas Health Science Center, Houston, TX 77030, USA
[3]École polytechnique fédérale de Lausanne, Lausanne, Switzerland
[4]Inpher, EPFL Innovation Park Bàtiment A, 3rd Fl, 1015 Lausanne, Switzerland
[5]CEA, LIST, 91191 Gif-sur-Yvette Cedex, France
[6]Department of Mathematical Sciences, Seoul National University, Seoul 08826, Republic of Korea
[7]Crypto Lab Inc., Seoul 08826, Republic of Korea
[8]Zama, Paris, France and imec-COSIC, KU Leuven, Leuven, Belgium
[9]West Coast Head of Research Science, Facebook AI Research (FAIR), Seattle, Washington
[10]University of Pennsylvania, Philadelphia, PA 19104, USA
[11]UCSD Health Department of Biomedical Informatics, University of California, San Diego, San Diego, CA 92093, USA
[12]National Institutes of Health (NIH) - National Human Genome Research Institute, Bethesda, MD 20892, USA
[13]Samsung SDS, Seoul, Republic of Korea
[14]Department of Computer Science and Engineering, Seoul National University, Seoul 08826, Republic of Korea
[15]Center for Secure Artificial intelligence For hEalthcare (SAFE), School of Biomedical Informatics, University of Texas Health Science Center, Houston, TX 77030, USA
[16]These authors contributed equally
[17]Lead contact
*Correspondence: arif.o.harmanci@uth.tmc.edu (A.O.H.), xiaoqian.jiang@uth.tmc.edu (X.J.)
https://doi.org/10.1016/j.cels.2021.07.010

SUMMARY

Genotype imputation is a fundamental step in genomic data analysis, where missing variant genotypes are predicted using the existing genotypes of nearby "tag" variants. Although researchers can outsource genotype imputation, privacy concerns may prohibit genetic data sharing with an untrusted imputation service. Here, we developed secure genotype imputation using efficient homomorphic encryption (HE) techniques. In HE-based methods, the genotype data are secure while it is in transit, at rest, and in analysis. It can only be decrypted by the owner. We compared secure imputation with three state-of-the-art non-secure methods and found that HE-based methods provide genetic data security with comparable accuracy for common variants. HE-based methods have time and memory requirements that are comparable or lower than those for the non-secure methods. Our results provide evidence that HE-based methods can practically perform resource-intensive computations for high-throughput genetic data analysis. The source code is freely available for download at https://github.com/K-miran/secure-imputation.

INTRODUCTION

Whole-genome sequencing (WGS) (Ng and Kirkness, 2010; Shendure et al., 2017) has become the standard technique in clinical settings for tailoring personalized treatments (Rehm, 2017) and in research settings for building reference genetic databases (Schwarze et al., 2018; 1000 Genomes Project Consortium, 2015; Chisholm et al., 2013). Technological advances in the last decade enabled a massive increase in the throughput of WGS methods (Heather and Chain, 2016), which provided the opportunity for population-scale sequencing (Goldfeder et al., 2017), where a large sample from a population is sequenced for studying ancestry and complex phenotypes (Lango Allen et al., 2010 throughout the article Locke et al., 2015), as well as rare (Agarwala et al., 2013; Gibson, 2012; Chen et al., 2019) and chronic diseases (Cooper et al., 2008). Although the price of sequencing has been decreasing, the sample sizes are increasing to accommodate the power necessary for new studies. It is anticipated that tens of millions of individuals will have access to their personal genomes in the next few years.

The increasing size of sequencing data creates new challenges for sharing, storage, and analyses of genomic data.

Among these, genomic data security and privacy have received much attention in recent years. Most notably, the increasing prevalence of genomic data, e.g., direct-to-consumer testing and recreational genealogy, makes it harder to share genomic data due to privacy concerns. Genotype data are very accurate in identifying the owner because of their high dimensionality, and leakage can cause concerns about discrimination and stigmatization (Nissenbaum, 2009). Also, the recent cases of forensic usage of genotype data are making it very complicated to share data for research purposes. The identification risks extend to family members of the owner, since a large portion of the genetic data are shared with relatives. Many attacks have been proposed on genomic data sharing models, where the correlative structure of the variant genotypes provides enough power to adversaries to make phenotype inference and individual re-identification possible (Nyholt et al., 2009). Therefore, it is of the utmost importance to ensure that genotype data are shared securely. There is a strong need for new methods and frameworks that will enable decreasing the cost and facilitate the analysis and management of genome sequencing.

One of the main techniques used for decreasing the cost of large-scale genotyping is *in silico* genotype imputation; i.e., measuring genotypes at a subsample of variants, e.g., using a genotyping array, and then utilizing the correlations among the genotypes of nearby variants (the variants that are close to each other in genomic coordinates) and imputing the missing genotypes using the sparsely genotyped variants (Howie et al., 2011; Das et al., 2018; Marchini and Howie, 2010). Imputation methods aim at capturing the linkage disequilibrium patterns on the genome. These patterns emerge because genomic recombination occurs at hotspots rather than at uniformly random positions along the genome. The genotyping arrays are designed around the idea of selecting a small set of ‘‘tag’’ variants, as small as 1% of all variants, that optimize the trade-off between cost and imputation accuracy (Hoffmann et al., 2011; Stram, 2004). Imputation methods learn the correlations among variant genotypes by using population-scale sequencing projects (Loh et al., 2016). In addition to filling in missing genotypes, the imputation process has many other advantages. Combining low-cost array platforms with computational genotype imputation methods decreases genotyping costs and increases the power of genome-wide association studies (GWASs) by increasing sample sizes (Tam et al., 2019). Accurate imputation can also greatly help with the fine-mapping of causal variants (Schaid et al., 2018) and is vital for meta-analysis of the GWAS (Evangelou and Ioannidis, 2013). Genotype imputation is now a standard and integral step in performing GWAS. Although imputation methods can predict only the variant genotypes that exist in the panels, the panels' sample sizes are increasing rapidly; e.g., in projects such as TOPMed (Taliun et al., 2019; TOPMed, 2016) will provide training data for imputation methods to predict rarer variant genotypes, and this can increase the sensitivity of GWAS.

Although imputation and sparse genotyping methods enable a vast decrease in genotyping costs, they are computationally very intensive and require management of large genotype panels and interpretation of the results (Howie et al., 2012). The imputation tasks can be outsourced to third parties, such as the Michigan Imputation Server, where users upload the genotypes (as a Variant Call Format, VCF, file) to a server that performs imputation internally using a large computing system. The imputed genotypes are then sent back to the user. However, there are major privacy (Naveed et al., 2015) and data security (Berger and Cho, 2019) concerns over using these services, since the genotype data are analyzed in plaintext format where any adversary who has access to the third party's computer can view, copy, or even modify the genotype data. As genotype imputation is one of the central initial steps in many genomic analysis pipelines, it is essential that the imputation be performed securely to ensure that these pipelines can be computed securely as a whole. For instance, although several secure methods for GWAS have been developed (Cho et al., 2018), if genotype imputation (a vital step in GWAS analyses) is not performed securely, it is not possible to make sure GWAS analysis can be performed securely.

In order to test the current state-of-the-art methodologies for benchmarking the feasibility of the cryptographic methods for genotype imputation, we organized the genotype imputation track in iDASH2019 Genomic Privacy Challenges. This track benchmarked more than a dozen methods on a small scale (supplemental information; Table S1) to rank the most promising approaches for secure genotype imputation. The methods developed by the top winning teams led us (organizers and contestants) to perform this study to report a more comprehensive analysis of the secure genotype imputation framework, including benchmarks with state-of-the-art methods. We developed and implemented several approaches for secure genotype imputation. Our methods make use of the homomorphic encryption (HE) formalism (Gentry, 2009) that provides mathematically provable, and potentially one of the strongest security guarantees for protecting genotype data while imputation is performed in an untrusted semi-honest environment. To include a comprehensive set of approaches, we focus on three state-of-the-art HE cryptosystems, namely, Brakerski/Fan-Vercauteren (BFV) (Brakerski, 2012; Fan and Vercauteren, 2012), Cheon-Kim-Kim-Song (CKKS) (Cheon et al., 2017), and the fully homomorphic encryption over the torus (TFHE) (Chillotti et al., 2020; Boura et al., 2018). In our HE-based framework, genotype data are encrypted by the data owner before outsourcing the data. After this point, the data always remain encrypted, i.e., encrypted in transit, in use, and at rest; it is never decrypted until the results are sent to the data owner. The strength of our HE-based framework stems from the fact that the genotype data remain encrypted even while the imputation is being performed. Hence, even if the imputation is outsourced to an untrusted third party, any semi-honest adversaries learn nothing from the encrypted data. This property makes the HE-based framework very powerful. For an untrusted third party who does not have access to the private key, the genotype data are indistinguishable from random noise (i.e., practically of no use) at any stage of the imputation process. Our HE framework provides the strongest form of security for outsourcing genotype imputation compared with any other approach under the same adversarial model.

HE-based frameworks have been deemed impractical since their inception. Therefore, in comparison with other cryptographically secure methods, such as multiparty computation (Cho et al., 2018) and trusted execution environments (Kockan et al., 2020), HE-based frameworks have received little attention.

Recent theoretical breakthroughs in the HE literature and a strong community effort (HES, n.d. 2020) have since rendered HE-based systems practical. However, many of these improvements are only beginning to be reflected in practical implementations and applications of HE algorithms. In this study, we provide evidence for the practicality of the HE formalism by building secure and ready-to-deploy methods for genotype imputation. We perform detailed benchmarking of the time and memory requirements of HE-based imputation methods and demonstrate the feasibility of large-scale secure imputation. In addition, we compared HE-based imputation methods with the state-of-the-art plaintext, i.e., non-secure, imputation methods, and we found comparable performance (with a slight decrease) in imputation accuracy with the benefit of total genomic data security.

We present HE-based imputation methods in the context of two main steps, as this enables a general modular approach. The first step is imputation model building, where imputation models are trained using the reference genotype panel with a set of tag variants (variant genotypes on an Illumina array platform) to impute the genotypes for a set of target variants, e.g., common variants in the 1,000 Genomes Project (1000 Genomes Project Consortium, 2015) samples. The second step is the secure imputation step, where the encrypted tag variant genotypes are used to predict the target genotypes (which are encrypted) by using the imputation models trained in the first step. This step, i.e., imputation model evaluation using the encrypted tag variant genotypes, is where the HE-based methods are deployed. In principle, the model training step needs to be performed only once when the tag variants do not change, i.e., the same array platform is used for multiple studies. Although these steps seem independent, model evaluation is heavily dependent on the representation and encoding of the genotype data, and the model complexity affects the timing and memory requirements of the secure outsourced imputation methods. However, our results suggest that linear models (or any other model that can be approximated by linear models) can be almost seamlessly trained and evaluated securely, where the model builders (1st step) and model evaluators (2nd step) can work independently. However, our results also show that there is an accompanying performance penalty, especially for the rare variants, in using these models, and we believe that new and accurate methods are needed to provide both privacy and imputation accuracy. It should be noted that the performance penalty stems not from HE-model evaluation but from the lower performance of plaintext models. We provide a pipeline that implements both model training and evaluation steps so that it can be run on any selection of tag variants. We make the implementations publicly available, so that they can be used as a reference by the computational genomics community.

RESULTS

We present the scenario and the setting for secure imputation and describe the secure imputation approaches we developed. Next, we present accuracy comparisons with the current state-of-the-art non-secure imputation methods and the time and memory requirements of the secure imputation methods. Finally, we present the comparison of the time and memory requirements of our secure imputation pipeline with the non-secure methods.

Genotype imputation scenario

Figure 1A illustrates the secure imputation scenario. A researcher genotypes a cohort of individuals by using genotyping arrays or other targeted methods, such as whole-exome sequencing, and calls the variants using a variant caller such as the Genome Analysis Toolkit, GATK (Depristo et al., 2011). After genotyping, the genotypes are stored in plaintext, i.e., unencrypted and not secure for outsourcing. Each variant genotype is represented by one of the three values $\{0, 1, 2\}$, where 0 indicates a homozygous reference genotype, 1 indicates a heterozygous genotype, and 2 indicates a homozygous alternate genotype. To secure the genotype data, the researcher generates two keys: a public key for encrypting the genotype data and a private key for decrypting the imputed data. The public key is used to encrypt the genotype data into ciphertext, i.e., random-looking data that contain the genotype data in a secure form. It is mathematically provable (i.e., equivalent to the hardness of solving the ring learning with errors, or RLWE, problem, Lyubashevsky et al., 2010) that the encrypted genotypes cannot be decrypted into plaintext genotype data by a third party without the private key, which is in the possession of only the researcher. Even if an unauthorized third party copies the encrypted data without authorization (e.g., hacking, stolen hard drives), they cannot gain any information from the data as they are essentially random noise without the private key. The security (and privacy) of the genotype data are therefore guaranteed, as long as the private key is not compromised. The security guarantee of the imputation methods is based on the fact that genotype data are encrypted in transit, during analysis, and at rest. The only plaintext data that are transmitted to the untrusted entity are the locations of the variants, i.e., the chromosomes and positions of the variants. Since the variant locations are publicly known for genotyping arrays, they should not leak any information. However, when the genotyping is performed by sequencing-based methods, the variant positions may leak information, as we discuss more in the next sections.

The encrypted genotypes are sent through a channel to the imputation service. The channel does not have to be secure against an eavesdropper because the genotype data are encrypted by the researcher. However, secure channels should be authenticated to prevent malicious man-in-the-middle attacks (Gangan, 2015). The encrypted genotypes are received by the imputation service, an honest-but-curious entity, i.e., they will receive the data legitimately and extract all the private information they can from the data. However, a privacy breach is impossible as the data are always encrypted when they are in the possession of the imputation service. Hence, the only reasonable action for the secure imputation server is to perform the genotype imputation and to return the data to the researcher. It is possible that the imputation server acts maliciously and intentionally returns bad-quality data to the researcher using badly calibrated models. However, it is economically or academically reasonable to assume that this is unlikely, since it would be easy to detect this behavior on the researcher's side and to advertise the malicious or low quality of the service to other researchers. Therefore, we assume that the secure server is semi-honest, and it performs the imputation task as accurately as possible. However, more complex malicious entities that perform complex attacks (e.g., slight biases in the models) are harder to detect. We treat these scenarios as out of scope of

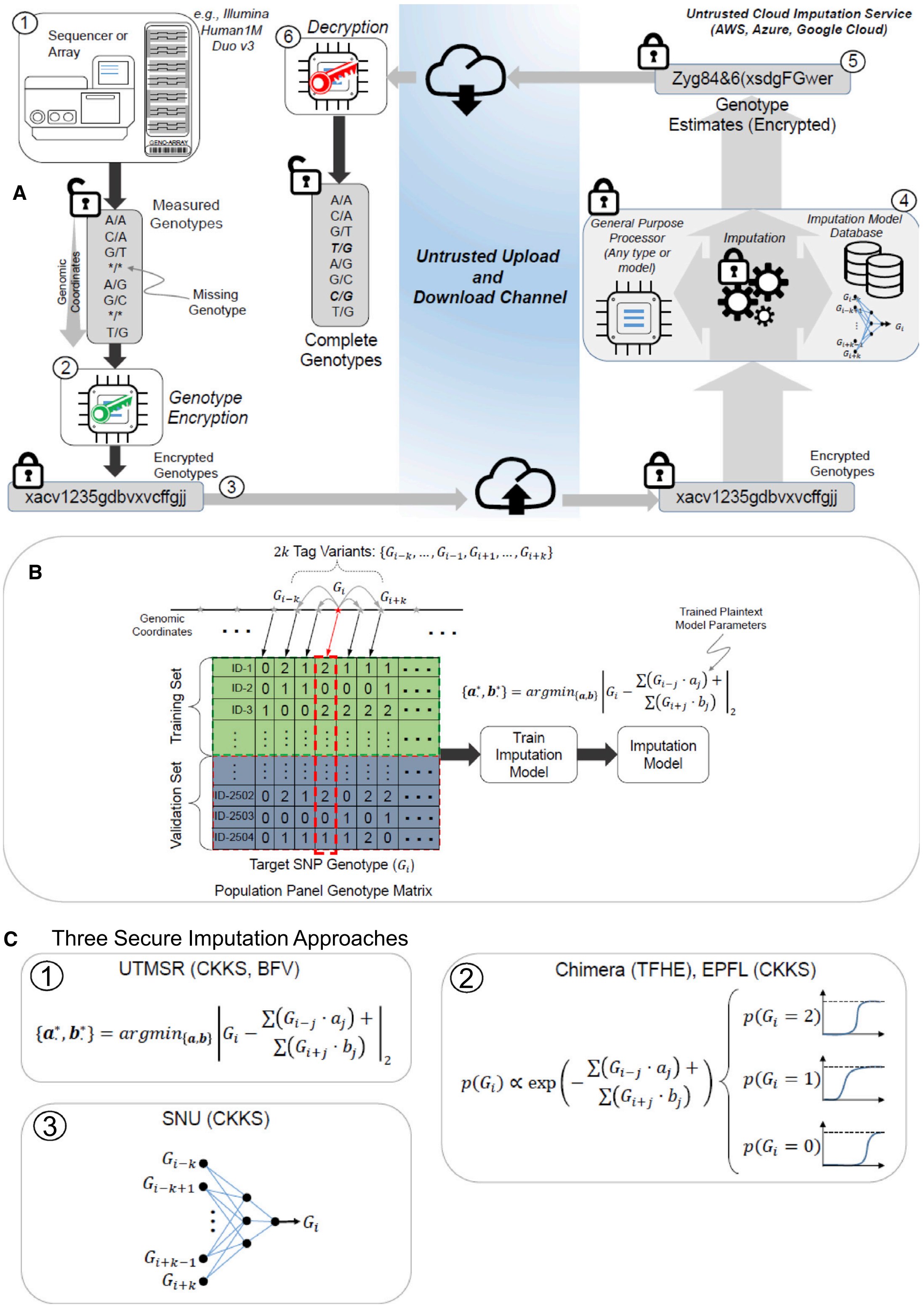

A
1 Sequencer or Array
e.g., Illumina Human1M Duo v3
Measured Genotypes
Genomic Coordinates
A/A C/A G/T */* A/G G/C */* T/G
Missing Genotype
2 Genotype Encryption
Encrypted Genotypes
xacv1235gdbvxvcffgjj
3
6 Decryption
A/A C/A G/T T/G A/G G/C C/G T/G
Complete Genotypes
Untrusted Upload and Download Channel
Untrusted Cloud Imputation Service (AWS, Azure, Google Cloud)
5 Zyg84&6(xsdgFGwer
Genotype Estimates (Encrypted)
4 General Purpose Processor (Any type or model)
Imputation
Imputation Model Database
Encrypted Genotypes
xacv1235gdbvxvcffgjj
B
2k Tag Variants: $\{G_{i-k}, \ldots, G_{i-1}, G_{i+1}, \ldots, G_{i+k}\}$
G_{i-k} G_i G_{i+k}
Genomic Coordinates
Trained Plaintext Model Parameters
Training Set
Validation Set
ID-1 0 2 1 2 1 1 1
ID-2 0 1 1 0 0 0 1
ID-3 1 0 0 2 2 2 2
ID-2502 0 2 1 2 0 2 2
ID-2503 0 0 0 0 1 0 1
ID-2504 0 1 1 1 1 2 0
$\{a_{\cdot}^{*}, b_{\cdot}^{*}\} = argmin_{\{a,b\}} \left| G_i - \begin{matrix} \Sigma(G_{i-j} \cdot a_j) + \\ \Sigma(G_{i+j} \cdot b_j) \end{matrix} \right|_2$
Train Imputation Model
Imputation Model
Target SNP Genotype (G_i)
Population Panel Genotype Matrix
C Three Secure Imputation Approaches
1 UTMSR (CKKS, BFV)
$\{a_{\cdot}^{*}, b_{\cdot}^{*}\} = argmin_{\{a,b\}} \left| G_i - \begin{matrix} \Sigma(G_{i-j} \cdot a_j) + \\ \Sigma(G_{i+j} \cdot b_j) \end{matrix} \right|_2$
2 Chimera (TFHE), EPFL (CKKS)
$p(G_i) \propto \exp\left(-\begin{matrix} \Sigma(G_{i-j} \cdot a_j) + \\ \Sigma(G_{i+j} \cdot b_j) \end{matrix}\right)$
$p(G_i = 2)$
$p(G_i = 1)$
$p(G_i = 0)$
3 SNU (CKKS)
G_{i-k} G_{i-k+1} G_{i+k-1} G_{i+k} G_i

Figure 1. Illustration of secure genotype imputation
(A) Illustration of the genotype imputation scenario. The incomplete genotypes are measured by genotyping arrays with missing genotypes (represented by stars). Encryption generates random-looking strings from the genotypes. At the server, encrypted genotypes are encoded, then they are used to compute the missing variant genotype probabilities. The encrypted probabilities are sent to the researcher, who decrypts the probabilities and identifies the genotypes with the highest probabilities (italic values).

(legend continued on next page)

our current study. Providing secure services against malicious entities is a worthwhile direction to explore for future studies.

After the receipt of the encrypted genotypes by the server, the first step is recoding of the encrypted data into a packed format (Figure S1) that is optimized for the secure imputation process. This step is performed to decrease time requirements and to optimize the memory usage of the imputation process. The data are coded to enable analysis of multiple genotypes in one cycle of the imputation process (Dowlin et al., 2017). The next step is the secure evaluation of the imputation models, which entails securely computing the genotype probability for each variant by using the encrypted genotypes. The variants received from the researcher are treated as tag variants whose genotypes are used as features in the imputation model to predict the "target" variants, i.e., the missing variants (Figure 1B). For each target variant, the corresponding imputation model uses the genotypes of the nearby tag variants to predict the target variant genotype in terms of genotype probabilities. In other words, we use a number of nearby tag variants to build an imputation model for the respective target variant such that the tag variants that are nearby (in genomic coordinates) are treated as the features for assigning genotype scores for the target variant. After the imputation is performed, the encrypted genotype probabilities are sent to the researcher. The researcher decrypts the genotype probabilities by using a private key. The final genotypes can be assigned using the maximum probability genotype estimate, i.e., by selecting the genotype with the highest probability for each variant.

Genotype imputation models

We provide five approaches implemented by four different teams. For simplicity of description, we refer to the teams as Chimera, EPFL, SNU, and UTHealth-Microsoft Research (UTMSR) (see STAR Methods). Among these, CKKS is used in three different approaches (EPFL-CKKS, SNU-CKKS, and UTMSR-CKKS), and BFV and TFHE are each utilized by separate approaches (UTMSR-BFV and Chimera-TFHE, respectively). The teams independently developed and trained the plaintext imputation models using the reference genotype panel dataset. For each target variant, the tag variants in the vicinity of the target variant are used for imputing the target variant, i.e., the tag variants in the vicinity are used as features in the imputation models. The chimera team trained a logistic regression model and the EPFL team trained a multinomial logistic regression model. (Figure S4; Tables S3, S4, and S7); the SNU team used a 1-hidden-layer neural network (Figures 1C, S2, and S3; Table S5); and the UTMSR team trained a linear regression model (Figures 1C and S5).

Genotype representation

All methods treat the genotypes as continuous predictions, except for the Chimera and SNU teams who utilized a one-hot encoding of the genotypes (see STAR Methods), e.g., $0 \rightarrow (1,0,0)$, $1 \rightarrow (0,1,0)$, and $2 \rightarrow (0,0,1)$.

Tag variant (feature) selection

The selection of the tag variants is important as these represent the features that are used for imputing each target variant. In general, we found that the models that use 30–40 tag variants provide optimal results (for the current array platform) in terms of imputation accuracy (Tables S2, S5, S6, and S8). As previous studies have shown, tag variant selection can provide an increase in imputation accuracy (Yu and Schaid, 2007). Finally, we observed a general trend of linear scaling with the number of target variants (as shown in Figure S6 and other Tables S1–S9). This provides evidence that there is minimal extra overhead (in addition to the linear increasing sample size) for scaling to genome-wide and population-wide computations.

Training and secure evaluation of models

We present the accuracy comparison results further on. We include extended discussion of the specific ideas used for training and for secure evaluation of the genotype imputation models in supplemental information.

Accuracy comparisons with the non-secure methods

We first analyzed the imputation accuracy of the secure methods with their plaintext (non-secure) counterparts that are the most popular state-of-the-art imputation methods. We compared secure imputation methods with IMPUTE2 (Howie et al., 2009), Minimac3 (Das et al., 2016) (and Minimac4, which is an efficient re-implementation of Minimac3), and Beagle (Browning et al., 2018) methods. These plaintext methods utilize Hidden Markov models (HMMs) for genotype imputation (see STAR Methods). The population panels and the pre-computed estimates of the recombination frequencies are taken as input to the methods. Each method is set to provide a measure of genotype probabilities, in addition to the imputed genotype values.

To perform comparisons in a realistic setting, we used the variants on the Illumina Duo 1M version 3 array platform (Johnson et al., 2013). This is a popular array platform that covers more than 1.1 million variants and is used by population-scale genotyping studies such as HAPMAP (Belmont et al., 2003). We extracted the genotypes of the variants that were probed by this array platform and overlapped with the variants identified by the 1,000 Genomes Project population panel of 2,504 individuals. For simplicity of comparisons, we focused on chromosome 22. The variants that are probed by the array are treated as the tag variants that are used to estimate the target variant genotypes. The target variants are defined as variants on chromosome 22 whose allele frequency is greater than 5% as estimated by the 1,000 Genomes Project (1000 Genomes Project Consortium, 2015). We used the 16,184 tag variants and 80,882 common target variants. Then, we randomly divided the 2,504 individuals into a training genotype panel of 1,500 samples and a testing panel of 1,004 samples. The training panel is used as the input to the plaintext methods (i.e., IMPUTE2, Minimac3-4, and Beagle) and also for building the plaintext imputation models of the secure methods. Each method is then used to impute the

(B) Building of the plaintext model for genotype imputation. The server uses a publicly available panel to build genotype estimation models for each variant. The models are stored in the plaintext domain. The model in the current study is a linear model where each variant genotype is modeled using genotypes of variants within a *k* variant vicinity of the target variant.
(C) The plaintext models implemented under the secure frameworks.

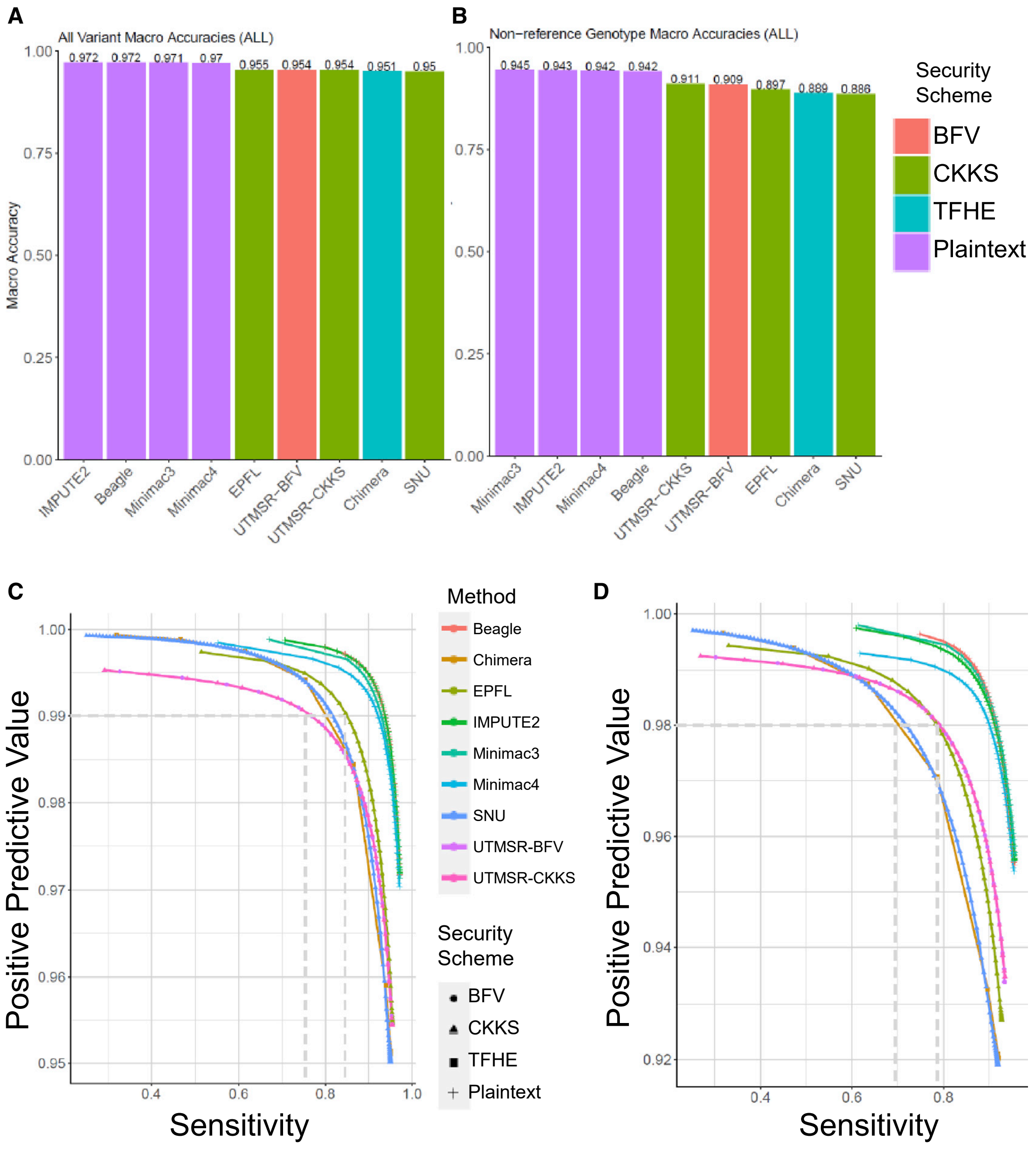
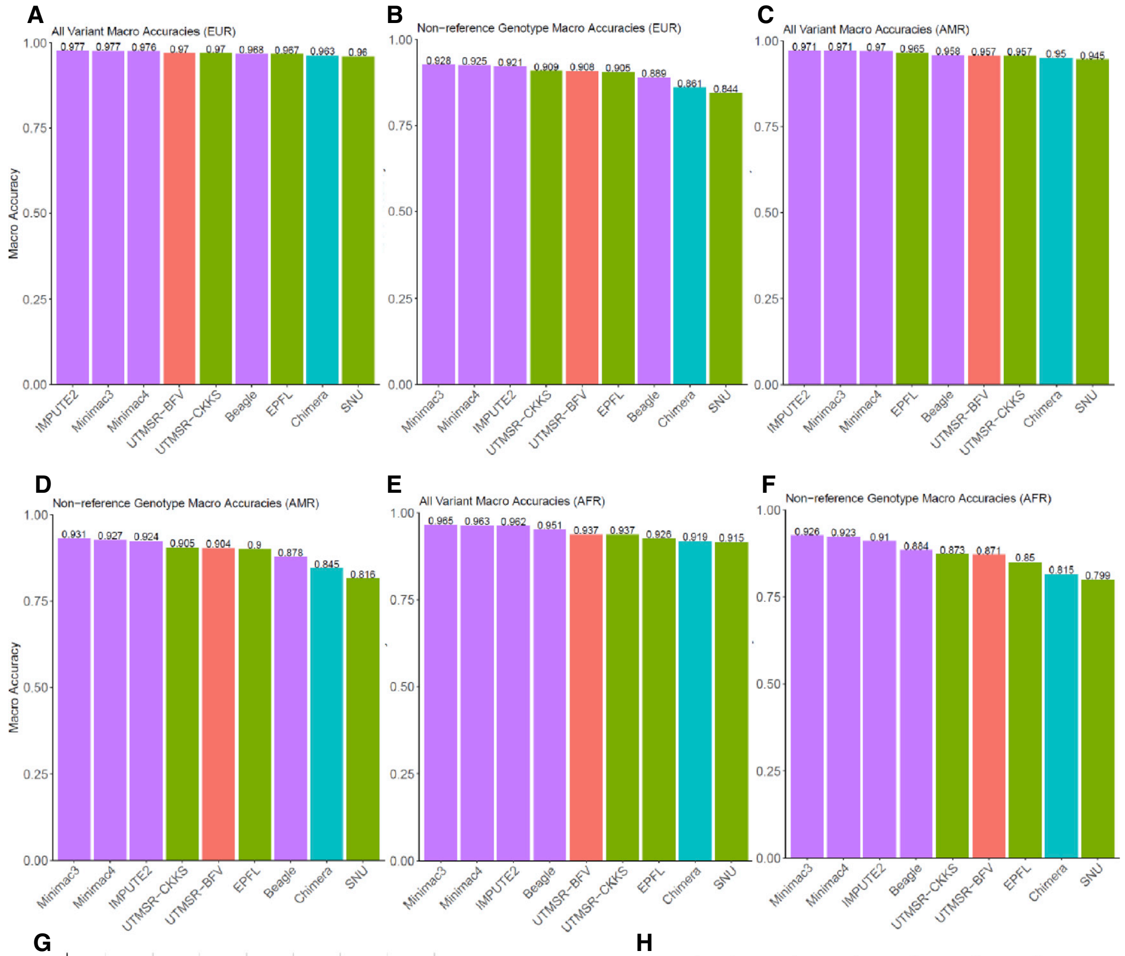


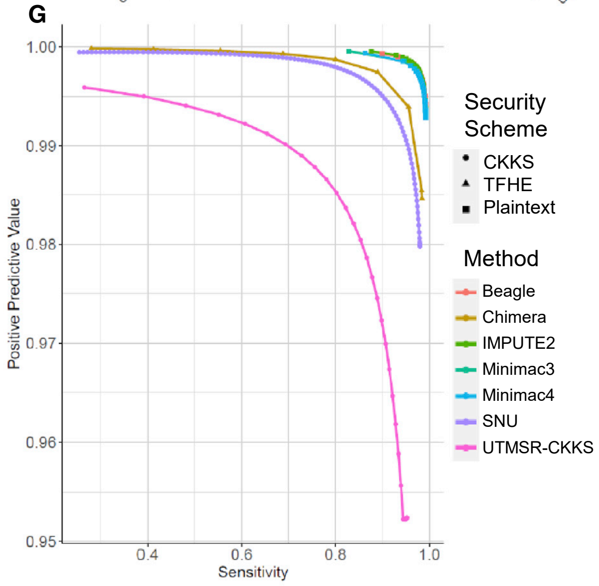
Figure 2. Accuracy benchmark

(A–D) Accuracy for all genotypes (A) and the non-reference genotypes (B) are shown for each method (x axis). The average accuracy value is shown at the top of each bar for comparison. Precision-recall curves are plotted for all genotypes (C) and the non-reference genotypes (D). Plaintext indicates the non-secure methods.

target variants using the tag variants. Figure 2A shows the comparison of genotype prediction accuracy computed over all the predictions made by the methods. The non-secure methods show the highest accuracy among all the methods. The secure methods exhibit very similar accuracy, whereas the closest method follows with only a 2%–3% decrease in accuracy. To understand the differences between the methods, we also computed the accuracy of the non-reference genotype predictions (see STAR Methods; Figure 2B). The non-secure methods show slightly higher accuracy compared with the secure methods. These results indicate that the proposed secure methods provide perfect data privacy at the cost of a slight decrease in imputation accuracy.

Next, we assessed whether the genotype probabilities (or scores) computed from the secure methods provide meaningful measures for choosing reliably imputed genotypes. For this, we calculated the sensitivity and the positive predictive value (PPV) of the imputed genotypes whose scores exceed the cutoff (see STAR Methods). To analyze how cutoff selections affect the accuracy metrics, we shifted the cutoff (swept the cutoff over the range of genotype scores) so that the accuracy is computed for the most reliable genotypes (high cutoff) and for the most





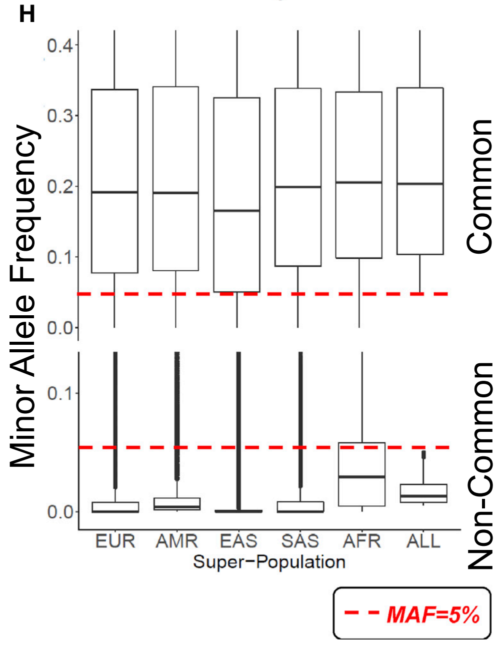


Figure 3. Genotype imputation accuracy benchmarking for stratified populations
(A–H) The population stratification of the accuracy is shown for EUR all genotypes (A) and non-ref genotypes (B), AMR all (C), and non-ref (D) genotypes, and AFR all (E), and non-ref genotypes.

(legend continued on next page)

inclusive genotypes (low cutoff). We then plotted the sensitivity versus the PPV (Figure 2C). Compared with the secure methods, the non-secure methods generally show higher sensitivity at the same PPV. However, secure methods can capture more than 80% of the known genotypes with 98% accuracy. The same results hold for the non-reference genotypes' prediction accuracy (Figure 2D). These results indicate that secure genotype predictions can be filtered by setting cutoffs to improve accuracy.

We also evaluated the population-specific effects on imputation accuracy. For this, we divided the testing panel into three populations—210 European (EUR), 135 American (AMR), and 272 African (AFR) samples—as provided by the 1,000 Genomes Project. The training panel yielded 389 AFR, 212 AMR, and 293 EUR samples. Figures 3A and 3B show genotype and non-reference genotype accuracy for EUR, respectively. We observed that the non-secure and secure methods are similar in terms of accuracy. We observed that the secure CKKS (UTMSR-CKKS) scheme with a linear prediction model outperformed Beagle in the EUR population, with marginally higher accuracy. We observed similar results for AMR populations where the non-secure methods performed at the top and secure methods showed very similar but slightly lower accuracy (Figures 3C and 3D). For AFR populations, the non-reference genotype prediction accuracy is lower for all the methods (Figures 3E and 3F). This is mainly rooted in the fact that the African populations show distinct properties that are not yet well characterized by the 1,000 Genomes Panels. We expect that the larger panels can provide better imputation accuracy.

To further investigate the nature of the imputation errors, we analyzed the characteristics of imputation errors of each method by computing the confusion matrices (Figure S7). We found that the most frequent errors are made when the real genotype is heterozygous, and the imputed genotype is a homozygous reference genotype. The pattern holds predominantly in secure and non-secure methods, although the errors are slightly lower, as expected, for the non-secure methods. Overall, these results indicate that secure imputation models can provide genotype imputations comparable with their non-secure counterparts.

To test the performance of the methods on rare variants, we focused on the 117,904 variants whose minor allele frequency (MAF) is between 0.5% and 5%. These variants represent harder to impute variants since they are much less represented compared with the common variants. The results show that the vicinity-based approaches that our methods use show a clear decrease in performance compared with the HMM-based approaches (Figure 3G). This is expected since our approaches depend heavily on the existence of well-represented training datasets. In the rare variants, however, the number of training examples for the non-reference genotypes goes as low as 1 or 2 examples over 1,000 individuals. That is why we observed a substantial decrease in the imputation power in our methods. Interestingly, we observed that the more complex methods (Chimera's logistic regression and SNU's neural network approach) provided comparably better accuracy than the ordinary linear model, which suggests that the more complex vicinity-based models can perform more accurate imputation for rare variants. In summary to this comparison, the rare variants represent challenging cases and a limitation for vicinity-based secure approaches.

It should be noted that a substantial portion of the rare variants are shown to be population specific (Bomba et al., 2017). To test for this, we analyzed the population specificity of the variants by computing the population-specific AF of these variants. We observed that most of the rare variants show enrichment in the African populations (Figure 3H) with a median MAF of around 2%–3% for AFR. Compared with the rare variants, the common variants showed a much more frequent and more uniform representation among the populations. These results highlight that rare variants can potentially be more accurately imputed using population-specific panels, which is in concordance with previous studies (Kowalski et al., 2019). Finally, from the perspective of downstream analyses, such as GWAS, high allele frequency variants are much more useful, since even the highly powered GWAS studies perform stringent MAF cutoffs at 2%–3% to ensure that the causal variants are not false positives (Sung et al., 2018).

Timing and memory requirements of secure imputation methods

One of the main critiques of HE methods is that they are impractical due to memory and time requirements. Therefore, we believe that the most important challenge is to make HE methods practical in terms of memory and time. To assess and demonstrate the practicality of the secure methods, we performed a detailed analysis of the time and memory requirements of secure imputation methods. We divided the imputation process into four steps (key generation, encryption, secure model evaluation, and decryption), and we measured the time and the overall memory requirements. Figure 4A shows the detailed time requirements for each step. In addition, we studied the scalability of secure methods. For this, we report the time requirements for 20,000 (20K), 40,000 (40K), and 80,000 (80K) target variants to present how the time requirements scale with the number of target variants. The secure methods spend up to 10 ms for key generation. In the encryption step, all methods were well below 2 s. The most time-consuming step of evaluation took less than 10 s, even for the largest set of 80K variants. Decryption, the last step, took less than 2 s. Except for the key generation and encryption, all methods exhibited linear scaling with the increasing number of target variants. Overall, the total time spent in secure model evaluation took less than 25 s (Figure 4B). This could be ignored when compared with the total time requirements of the non-secure imputation. Assuming that time usage scales linearly with the number of target variants (Figure 3A), 4 million variants can be evaluated in approximately 1,250 s, which is less than half an hour. In other terms, secure evaluation is approximately 312 μs. per variant per 1,000 individuals $(25\,\mathrm{s} \times 1,000\ \mathrm{individuals})$. It can be decreased even further by scaling

(F). Precision-recall curve for rare variants (G). The boxplots illustrate the super-population-specific minor allele frequency distribution (y axis) for the common (top) and un-common variants (bottom)

(H). ALL indicates the MAF distribution for all populations. The center and the two ends of the boxplots show the median and 25%–75% values of the MAF distributions.

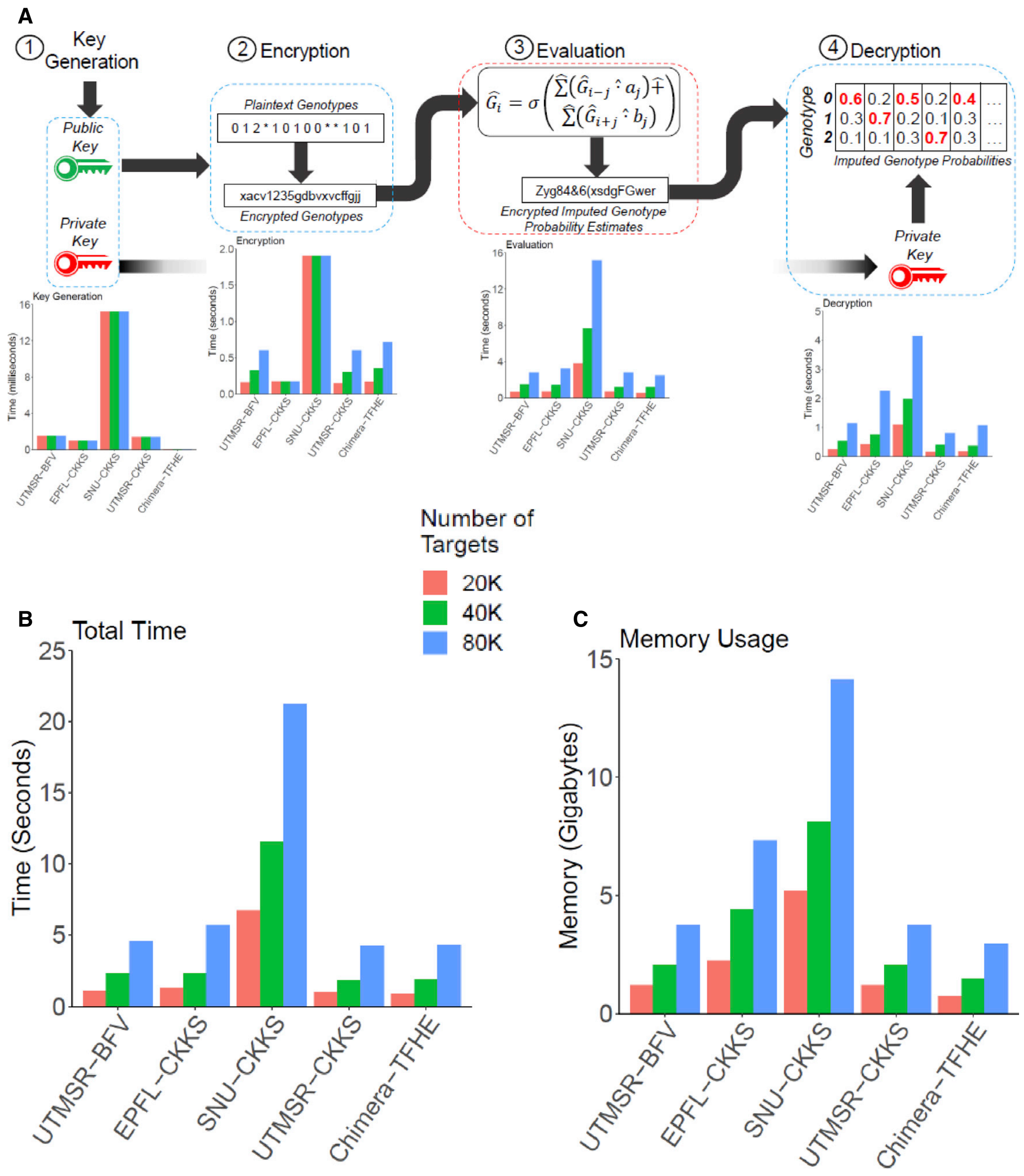


Figure 4. Memory and time requirements of the secure methods
(A–C) Each method is divided into 4 steps: (1) key generation, (2) encryption, (3) evaluation, and (4) decryption. The bar plots show the time requirements (A) using 20K, 40K, and 80K target variant sets. The aggregated time (B) and the maximum memory usage of the methods are also shown (C).

to a higher number of CPUs (i.e., cores on local machines or instances on cloud resources). In terms of memory usage, all methods required less than 15 gigabytes of main memory, and three of the five approaches required less than 5 gigabytes (Figure 4C). These results highlight the fact that secure methods could be deployed on even the commodity computer systems. The training of the methods on rare variants were performed to ensure the assigned scores are best tuned for the unbalanced training data in rare variants. The Chimera and SNU teams (best performing methods) have a diverse range of requirements for secure evaluation where the neural network approach (SNU) requires high resources, whereas the logistic regression approach has much more practicable resource requirements (Tables S9 and S10).

Resource usage comparison between secure and non-secure imputation methods

An important aspect of practicality is whether the methods are adaptable to different tag variants. This issue arises when a new array platform is used for genotyping tag variants with a

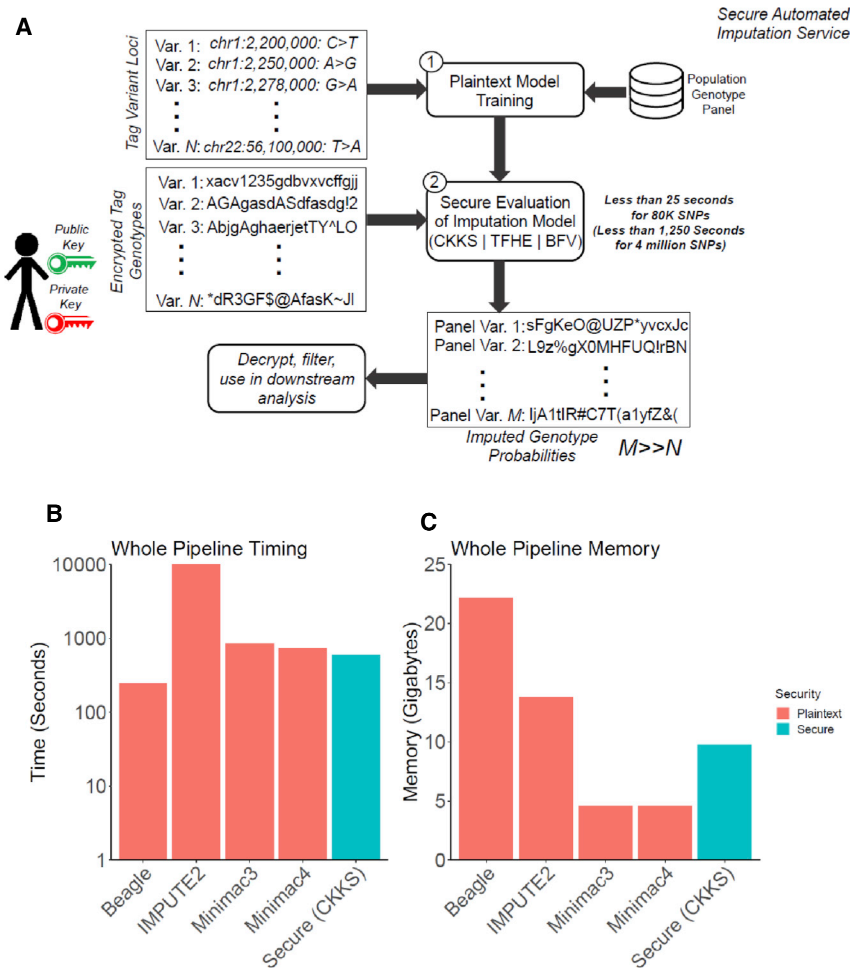


Figure 5. Comparison of time and memory requirements of methods

(A– C) The secure outsourced imputation service (A), time (B), and memory requirements (C) are illustrated in the bar plots where colors indicate security context. The y axis shows the time (in seconds) and main memory (in gigabytes) used by each method to perform the imputation of the 80K variants where the secure outsourced method includes the plaintext model training and secure model evaluation steps.

new set of tag variant loci. In this case, the current security framework requires that the plaintext models must be re-parametrized, and this may require a large amount of time and memory. To evaluate this, we optimized the linear models for the UTMSR-CKKS approach and measured the total time (training and evaluation) and the memory for the target variant set.

In order to make the comparisons fair with the HMM-based methods, we included the rare variants and common variants in this benchmark where the variants with MAF greater than 0.5% are used. In total, we used the 200,976 target variants in this range. In this way, we believe that we perform a fair comparison of resource usage with other non-secure methods. We assumed that the training and secure evaluation would be run sequentially, and we measured the time requirement of the secure approach by summing the time for key generation, encryption, secure evaluation, decryption, and the time for training. For memory, we computed the peak memory required for training and the peak memory required for secure evaluation. These time and memory requirements provided us with an estimate of the resources used by the secure pipeline (Figure 5A) that can be fairly compared with the non-secure methods.

We measured the time and memory requirements of all the methods by using a dedicated computer cluster to ensure resource requirements are measured accurately (see STAR Methods). For IMPUTE2, there was no option for specifying multiple threads. Hence, we divided the sequenced portion of chromosome 22 into 16 regions and imputed variants in each region in parallel using IMPUTE2, as instructed by the manual, i.e., we ran 16 IMPUTE2 instances in parallel to complete the computation. We then measured the total memory required by all 16 runs and used this as the memory requirement by IMPUTE2. We used the maximum time among all the 16 runs, as the time requirements by parallelized IMPUTE2. Beagle, Minimac3, and Minimac4 were run with 16 threads, as this option was available in the command line. In addition, Minimac4 requires model parametrization and preprocessing of the reference panel, which requires large CPU time. Therefore, we included this step in the timing requirements. Figures 5B and 5C show the time and memory requirements, respectively, of the three non-secure approaches and our secure method. The results show that the secure pipeline provides competitive timing (2nd fastest after Beagle) and memory requirements (3rd in terms of least usage after Minimac3 and Minimac4). Our results also show that Minimac3/Minimac4 and our secure approach provided a good trade-off between memory and timing, because Beagle and IMPUTE2 exhibit the highest time or highest memory requirements compared with other methods.

We also compared the secure models and found that different secure models exhibit diverse accuracy depending on allele frequency and position of variants (supplemental information).

DISCUSSION

We presented fully secure genotype imputation methods that can practically scale to genome-wide imputation tasks by using efficient HE techniques where the data are encrypted in transit, in analysis, and at rest. This is a unique aspect of the HE-based frameworks because, when appropriately performed, encryption is one of the few approaches that are recognized at the legislative level as a way of secure sharing of biomedical data, e.g., by HIPAA (Wilson, 2006) and partially by GDPR (Hoofnagle et al., 2019).

Our study was enabled by several key developments in the fields of genomics and computer science. First, the recent theoretical breakthroughs in the HE techniques have enabled massive increases in the speed of secure algorithms. Although much of the data science community still regards HE as a theoretical and not-so-practical framework, the reality is far from this image. We hope that our study can provide a reference for the development of privacy-aware and fully secure approaches that employ HE. Second, the amount of genomic data have increased several orders of magnitude in recent years. This provides enormous genotype databases where we can train the imputation models and test them in detail before implementing them in secure evaluation frameworks. Another significant development is the recent formation of genomic privacy communities and alliances, i.e., Global Alliance for Genomic Health (GA4GH), where researchers build interdisciplinary approaches for developing privacy-aware methods. For example, our international study stemmed from the 2019 iDASH Genomic Privacy Challenge. We firmly believe that these communities will help bring together further interdisciplinary collaborations for the development of secure genomic analysis methods.

The presented imputation methods train an imputation model for each target variant. Our approach handles millions of models, i.e., parameters. Unlike the HMM models that can adapt seamlessly to a new set of tag variants (i.e., a new array platform), our approaches need to be retrained when the tag variants are updated. We expect that the training can be performed a-priori for a new genotyping array and that it can be reused in the imputation. The decoupling of the (1) plaintext training and (2) secure evaluation steps is very advantageous, because plaintext training can be independently performed at the third party without the need to wait for the data to arrive. This way, the users would have to accrue only the secure evaluation time, that is, as our results show, much smaller compared with the time requirements of the non-secure models, as small as 312 μs per variant per 1,000 individuals. Nevertheless, even with the training, our results show that the secure imputation framework can train and evaluate in run times comparable with plaintext (non-secure) methods. In the future, we expect many optimizations can be introduced to the models we presented. For example, we foresee that the linear model training can be replaced with more complex feature selection and training methods. Deep neural networks are potential candidates for imputation tasks, as they can be trained for learning the complex haplotype patterns to provide better imputation accuracy (Das et al., 2018). With the introduction of the graphical processing units (GPUs) on the cloud, these models can be trained and evaluated securely and efficiently. It is, however, important to be thorough about the security of the data because, as we mentioned before, even the number of untyped target variants that the researcher sends to the server can leak some information about the datasets. These stealthy leakages highlight the importance of using semantic security approaches. It is important to note that the secure evaluation steps implemented in our study replicate the results of the plaintext models almost exactly, which indicates that "HE-conversion" does not accrue any performance penalty.

Our study aims to spearhead the feasibility of secure genotype imputation in a high-throughput manner. As such, there are currently numerous limitations that must be overcome in future studies (supplemental information). For example, our approaches provide suboptimal accuracy when compared with non-secure methods, especially for rare variants. As we mentioned earlier, we foresee that our methods can be optimized in numerous ways. For instance, it has been previously shown that the vicinity-based methods can make use of tag single nucleotide polymorphism (SNP) selection to increase accuracy (Yu and Schaid, 2007). We are also foreseeing that new methods can be adapted on the hard-to-impute regions (Duan et al., 2013; Chen and Shi, 2019) to provide higher accuracy for these regions with complex haplotype structures.

Finally, we believe that the multitude of models and the secure evaluation approaches that we presented here can help provide a much needed reference point for the development and improvement of the imputation methods. Moreover, the developed models can be easily adapted to solve other privacy-sensitive problems by using secure linear, logistic, and network model evaluations, such as the secure rare variant association tests (Wu et al., 2011). Therefore, we believe that our codebases represent an essential resource for the computational genomics community. We have organized the codebases to ensure that they can be most accessible to the users without the necessary cryptography expertise. We are hoping that our codebase can provide a central role in the development of a community (similar to *dynverse* (dynverse, n.d.) or TAPE (TAPE, n.d. 2019; Rao et al., 2019) repositories for trajectory inference and protein embedding, respectively) where users can use the developed methods and datasets for uniform benchmarking of their new imputation methods.

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
  - Variant and genotype datasets
  - Accuracy benchmark metrics
  - Micro-AUC accuracy statistics
  - Measurement of time and memory requirements
  - Secure methods
  - UTMSR-BFV and UTMSR-CKKS
  - Chimera-TFHE
  - EPFL-CKKS
  - SNU-CKKS
  - Non-secure methods
  - Beagle
  - IMPUTE2
  - Minimac3 and Minimac4

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.cels.2021.07.010.

ACKNOWLEDGMENTS

Authors thank the National Human Genome Research Institute (NHGRI) of National Institutes of Health for providing funding and support for iDASH Genomic Privacy challenges (R13HG009072). We also thank Luyao Chen for providing technical support to set up the computational environment for unified evaluation. X.J. is CPRIT Scholar in Cancer Research (RR180012), and he was supported in part by Christopher Sarofim Family Professorship, UT Stars award, UTHealth startup, the National Institutes of Health (NIH) under award number R13HG009072, R01GM114612, and the National Science Foundation (NSF) RAPID #2027790. L.O.-M. is supported by the NIH under award number R13HG009072 and R01GM114612 for this work. EPFL team is funded in part by the grant #2017-201 (DPPH) of the Swiss PHRT and by the grant #2018-522 (MedCo) of the Swiss PHRT and SPHN. I.C. has been supported in part by ERC Advanced Grant ERC-2015-AdG-IMPaCT, by the FWO under an Odysseus project GOH9718N and by the CyberSecurity Research Flanders with reference number VR20192203. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the ERC or FWO. D.K., S.H. and J.H.C. were supported by the Institute for Information & Communications Technology Promotion (IITP) Grant through the Korean Government (MSIT), (Development and Library Implementation of Fully Homomorphic Machine Learning Algorithms supporting Neural Network Learning over Encrypted Data), under Grant 2020-0-00840. M.K. was supported by the Settlement Research Fund (No. 1.200109.01) of UNIST (Ulsan National Institute of Science & Technology) and Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) (No.2020-0-01336, Artificial Intelligence graduate school support [UNIST]).

AUTHOR CONTRIBUTIONS

M.K., A.O.H., and X.J. designed the imputation scenario, implemented the evaluation metrics, conducted the benchmarking experiments with the baseline methods, and drafted the manuscript. M.K., A.O.H., X.J., Y. Song, and K.L. designed the baseline methods for UTMSR imputation pipelines. I.C., S.C., M.G., and N.G. trained and implemented the Chimera-TFHE pipeline and contributed the results to the manuscript. D.K., W.C., S.H., Y. Son, and J.H.C. trained and implemented the SNU-CKKS pipeline. Y.M., J.T.-P., D.F., J.-P.B., and J.-P.H. trained and implemented the EPFL-CKKS pipeline. H.S., L.O.-M., and X.J. oversaw the iDASH19 challenge, conceived the study, and edited the manuscript. All authors have read and approved the final manuscript.

DECLARATION OF INTERESTS

The authors declare no competing interests.

Received: August 25, 2020
Revised: April 21, 2021
Accepted: July 29, 2021
Published: August 30, 2021

REFERENCES

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

Agarwala, V., Flannick, J., Sunyaev, S.; GoT2D Consortium, and Altshuler, D. (2013). Evaluating empirical bounds on complex disease genetic architecture. Nat. Genet. *45*, 1418–1427.

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/standard/.

Albrecht, M.R., Player, R., and Scott, S. (2015). On the concrete hardness of learning with errors. J. Math. Cryptol. *9*, 169–203. http://www.degruyter.com/view/j/jmc.2015.9.issue-3/jmc-2015-0016/jmc-2015-0016.xml.

Belmont, J.W., Hardenbol, P., Willis, T.D., Yu, F., Yang, H., Ch'Ang, L.Y., Huang, W., Liu, B., Shen, Y., Tam, P.K.H., et al. (2003). The international hapmap project. Nature *426*, 789–796.

Berger, B., and Cho, H. (2019). Emerging technologies towards enhancing privacy in genomic data sharing. Genome Biol. *20*, 128.

Bomba, L., Walter, K., and Soranzo, N. (2017). The impact of rare and low-frequency genetic variants in common disease. Genome Biol. *18*, 77.

Boura, C., Gama, N., Georgieva, M., and Jetchev, D. (2018). Chimera: combining ring-lwe-based fully homomorphic encryption schemes, Technical report, Cryptology eprint Archive, Report 2018/758. https://eprint.iacr.org/2018/758.

Brakerski, Z. (2012). Fully homomorphic encryption without modulus switching from classical GapSVP. In Advances in Cryptology – Crypto 2012, R. Safavi-Naini and R. Canetti, eds. (Springer), pp. 868–886.

Browning, B.L., Zhou, Y., and Browning, S.R. (2018). A one-penny imputed genome from next-generation reference panels. Am. J. Hum. Genet. *103*, 338–348.

Chen, J., Harmanci, A.S., and Harmanci, A.O. (2019). Detecting and annotating rare variants. Encyclopedia of Bioinformatics and Computational Biology *3*, 388–399.

Chen, J., and Shi, X. (2019). Sparse convolutional denoising autoencoders for genotype imputation. Genes *10*, 652. https://doi.org/10.3390/genes10090652.

Cheon, J.H., Kim, A., Kim, M., and Song, Y. (2017). Homomorphic encryption for arithmetic of approximate numbers. In International Conference on the Theory and Application of Cryptology and Information Security (Springer), pp. 409–437.

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

Chisholm, J., Caulfield, M., Parker, M., Davies, J., and Palin, M. (2013). Briefing - Genomics England and the 100K Genome Project. Genomics Engl.

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

Cooper, J.D., Smyth, D.J., Smiles, A.M., Plagnol, V., Walker, N.M., Allen, J.E., Downes, K., Barrett, J.C., Healy, B.C., Mychaleckyj, J.C., et al. (2008). Meta-analysis of genome-wide association study data identifies additional type 1 diabetes risk loci. Nat. Genet. *40*, 1399–1401.

Das, S., Abecasis, G.R., and Browning, B.L. (2018). Genotype imputation from large reference panels. Annu. Rev. Genomics Hum. Genet. *19*, 73–96.

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.

Depristo, M.A., Banks, E., Poplin, R., Garimella, K.V., Maguire, J.R., Hartl, C., Philippakis, A.A., Del Angel, G., Rivas, M.A., Hanna, M., et al. (2011). A framework for variation discovery and genotyping using next-generation DNA sequencing data. Nat. Genet. *43*, 491–498.

Dowlin, N., Gilad-Bachrach, R., Laine, K., Lauter, K., Naehrig, M., and Wernsing, J. (2017). Manual for using homomorphic encryption for bioinformatics. Proc. IEEE *105*, 552–567.

Duan, Q., Liu, E.Y., Croteau-Chonka, D.C., Mohlke, K.L., and Li, Y. (2013). A comprehensive SNP and indel imputability database. Bioinformatics *29*, 528–531. https://doi.org/10.1093/bioinformatics/bts724.

Evangelou, E., and Ioannidis, J.P. (2013). Meta-analysis methods for genome-wide association studies and beyond. Nat. Rev. Genet. *14*, 379–389.

Fan, J., and Vercauteren, F. (2012). Somewhat practical fully homomorphic encryption. IACR Cryptol. Eprint Arch. *2012*, 144.

Gangan, S. (2015). A review of man-in-the-middle attacks. arXiv http://arxiv.org/abs/1504.02115.

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing, STOC '09, pp. 169–178. http://doi.acm.org/10.1145/1536414.1536440.

Gibson, G. (2012). Rare and common variants: twenty arguments. Nat. Rev. Genet. *13*, 135–145.

dynverse. dynverse: benchmarking, constructing and interpreting single-cell trajectories. https://github.com/dynverse.

Goldfeder, R.L., Wall, D.P., Khoury, M.J., Ioannidis, J.P.A., and Ashley, E.A. (2017). Human genome sequencing at the population scale: a primer on high-throughput DNA sequencing and analysis. Am. J. Epidemiol. *186*, 1000–1009.

Heather, J.M., and Chain, B. (2016). The sequence of sequencers: the history of sequencing DNA. Genomics *107*, 1–8.

HES (2020). Homomorphic encryption standardization (HES). https://homomorphicencryption.org.

Hoffmann, T.J., Kvale, M.N., Hesselson, S.E., Zhan, Y., Aquino, C., Cao, Y., Cawley, S., Chung, E., Connell, S., Eshragh, J., et al. (2011). Next generation genome-wide association tool: design and coverage of a high-throughput European-optimized SNP array. Genomics *98*, 79–89.

Hoofnagle, C.J., van der Sloot, B., and Borgesius, F.Z. (2019). The European Union general data protection regulation: what it is and what it means. Inf. Commun. Technol. Law *28*, 65–98.

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.

Howie, B., Marchini, J., and Stephens, M. (2011). Genotype imputation with thousands of genomes. G3 (Bethesda) *1*, 457–470.

Howie, B.N., Donnelly, P., and Marchini, J. (2009). A flexible and accurate genotype imputation method for the next generation of genome-wide association studies. PLoS Genet. *5*, e1000529.

Johnson, E.O., Hancock, D.B., Levy, J.L., Gaddis, N.C., Saccone, N.L., Bierut, L.J., and Page, G.P. (2013). Imputation across genotyping arrays for genome-wide association studies: assessment of bias and a correction strategy. Hum. Genet. *132*, 509–522.

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.

Kowalski, M.H., Qian, H., Hou, Z., Rosen, J.D., Tapia, A.L., Shan, Y., Jain, D., Argos, M., Arnett, D.K., Avery, C., et al. (2019). Use of >100,000 NHLBI Trans-Omics for Precision Medicine (TOPMed) Consortium whole genome sequences improves imputation quality and detection of rare variant associations in admixed African and Hispanic/Latino populations. PLoS Genet. *15*, e1008500.

Lango Allen, H.L., Estrada, K., Lettre, G., Berndt, S.I., Weedon, M.N., Rivadeneira, F., Willer, C.J., Jackson, A.U., Vedantam, S., Raychaudhuri, S., et al. (2010). Hundreds of variants clustered in genomic loci and biological pathways affect human height. Nature *467*, 832–838.

Locke, A.E., Kahali, B., Berndt, S.I., Justice, A.E., Pers, T.H., Day, F.R., Powell, C., Vedantam, S., Buchkovich, M.L., Yang, J., et al. (2015). Genetic studies of body mass index yield new insights for obesity biology. Nature *518*, 197–206.

Loh, P.R., Danecek, P., Palamara, P.F., Fuchsberger, C., A Reshef, Y., K Finucane, H., Schoenherr, S., Forer, L., McCarthy, S., Abecasis, G.R., et al. (2016). Reference-based phasing using the Haplotype Reference Consortium panel. Nat. Genet. *48*, 1443–1448.

Lyubashevsky, V., Peikert, C., and Regev, O. (2010). On ideal lattices and learning with errors over rings. Journal of the ACM *60*, 1–25.

Marchini, J., and Howie, B. (2010). Genotype imputation for genome-wide association studies. Nat. Rev. Genet. *11*, 499–511.

Naveed, M., Ayday, E., Clayton, E.W., Fellay, J., Gunter, C.A., Hubaux, J.P., Malin, B.A., and Wang, X. (2015). Privacy in the genomic era. ACM Comput. Surv. *48*, 1–44.

Ng, P.C., and Kirkness, E.F. (2010). Whole genome sequencing. In Genetic Variation (Springer), pp. 215–226.

Nissenbaum, H. (2009). Privacy in Context: Technology, Policy, and the Integrity of Social Life (Stanford University Press).

Nyholt, D.R., Yu, C.E., and Visscher, P.M. (2009). On Jim Watson's APOE status: genetic information is hard to hide. Eur. J. Hum. Genet. *17*, 147–149.

Rao, R., Bhattacharya, N., Thomas, N., Duan, Y., Chen, X., Canny, J., Abbeel, P., and Song, Y.S. (2019). Evaluating protein transfer learning with TAPE. bioRxiv. https://doi.org/10.1101/676825.

Rehm, H.L. (2017). Evolving health care through personal genomics. Nat. Rev. Genet. *18*, 259–267.

Schaid, D.J., Chen, W., and Larson, N.B. (2018). From genome-wide associations to candidate causal variants by statistical fine-mapping. Nat. Rev. Genet. *19*, 491–504.

Schwarze, K., Buchanan, J., Taylor, J.C., and Wordsworth, S. (2018). Are whole-exome and whole-genome sequencing approaches cost-effective? A systematic review of the literature. Genet. Med. *20*, 1122–1130.

Shendure, J., Balasubramanian, S., Church, G.M., Gilbert, W., Rogers, J., Schloss, J.A., and Waterston, R.H. (2017). DNA sequencing at 40: past, present and future. Nature *550*, 345–353.

Stram, D.O. (2004). Tag SNP selection for association studies. Genet. Epidemiol. *27*, 365–374.

Sung, Y.J., Winkler, T.W., de las Fuentes, L., Bentley, A.R., Brown, M.R., Kraja, A.T., Schwander, K., Ntalla, I., Guo, X., Franceschini, N., et al. (2018). A large-scale multi-ancestry genome-wide study accounting for smoking behavior identifies multiple significant loci for blood pressure. Am. J. Hum. Genet. *102*, 375–400.

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. (2019). Sequencing of 53,831 diverse genomes from the NHLBI TOPMed Program. bioRxiv. https://doi.org/10.1101/563866.

TAPE. (2019). Tasks assessing protein embeddings (TAPE), a set of five biologically relevant semi-supervised learning tasks spread across different domains of protein biology. https://github.com/songlab-cal/tape.

Tam, V., Patel, N., Turcotte, M., Bossé, Y., Paré, G., and Meyre, D. (2019). Benefits and limitations of genome-wide association studies. Nat. Rev. Genet. *20*, 467–484.

TOPMed. (2016). NHLBI trans-omics for precision medicine whole genome sequencing program. https://www.nhlbiwgs.org/.

Wilson, J.F. (2006). Health Insurance Portability and Accountability Act privacy rule causes ongoing concerns among clinicians and researchers. Ann. Intern. Med. *145*, 313–316.

Wu, M.C., Lee, S., Cai, T., Li, Y., Boehnke, M., and Lin, X. (2011). Rare-variant association testing for sequencing data with the sequence kernel association test. Am. J. Hum. Genet. *89*, 82–93.

Yu, Z., and Schaid, D.J. (2007). Methods to impute missing genotypes for population data. Hum. Genet. *122*, 495–504.

STAR★METHODS

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
The chromosome 22 genotype calls the 2,504 individuals in The 1000 Genomes Project's 3rd phase.	The 1000 Genomes Consortium	ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/release/20130502/ALL.chr22.phase3_shapeit2_mvncall_integrated_v5a.20130502.genotypes.vcf.gz
Illumina Duo version 3 genotyping array documentation	Illumina Inc. Web Site	https://support.illumina.com/downloads/human1m-duo_v3-0_product_files.html
Source Data for Figures 1, 2, 3, 4, and 5 and supplemental information	This work	https://doi.org/10.5281/zenodo.4947832
Software and algorithms		
R Statistical Computing Platform	The R Foundation	https://www.r-project.org/
Source Code and Documentation for Secure Imputation Models	This work	https://doi.org/10.5281/zenodo.4948000
Source Code for generating Figures 1, 2, 3, 4, and 5 and the Figures S1–S8	This work	https://doi.org/10.5281/zenodo.4947832

We present and describe the data sources, accuracy metrics, and non-secure imputation method parameters. The detailed methods are presented in the supplemental information.

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources and reagents should be directed to and will be fulfilled by the lead contact, Arif Harmanci (arif.o.harmanci@uth.tmc.edu).

Materials availability

This study did not generate new materials.

Variant and genotype datasets

All the tag and target variant loci, and the genotypes are collected from the public resources. We downloaded the Illumina Duo 1M version 3 variant loci from the array's specification at the Illumina web site (https://support.illumina.com/downloads/human1m-duo_v3-0_product_files.html). The file was parsed to extract the variants on chromosome 22, which yielded 17,777 variants. We did not use the CNVs and indels while filtering the variants and we focused only on the single nucleotide polymorphims (SNPs). We then intersected these variants with the 1000 Genomes variants on chromosome 22 to identify the array variants that are detected by the 1000 Genomes Project. We identified 16,184 variants from this intersection. This variant set represents the tag variants that are used to perform the imputation. The phased genotypes on chromosome 22 for the 2,504 individuals in the 1000 Genomes Project are downloaded from the NCBI portal (ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/release/20130502/ALL.chr22.phase3_shapeit2_mvncall_integrated_v5a.20130502.genotypes.vcf.gz). We filtered out the variants for which the allele frequency reported by the 1000 Genomes Project is less than 5%. After excluding the tag variants on the array platform, we identified 83,072 target variants that are to be used for imputation. As the developed secure methods use vicinity variants, the variants at the ends of the chromosome are not imputed. We believe this is acceptable because these variants are located very close to the centromere and at the very end of the chromosome. After filtering the non-imputed variants, we focused on the 80,882 variants that were used for consistent benchmarking of all the secure and non-secure methods.

Accuracy benchmark metrics

We describe the genotype level and variant level accuracy. For each variant, we assign the genotype with the highest assigned genotype probability. The variant level accuracy is the average variant accuracy where each variant's accuracy is estimated based on how well these imputed genotypes of the individuals match the known genotypes:

$$\text{Variant Acc.} = \left(\frac{1}{\#\text{ of Variants}}\right) \times \sum_i \left(\frac{\#\text{ Correctly Imputed Individuals for Variant } i}{\#\text{ of Individuals for Variant } i}\right).$$

Variant level accuracy is also referred to as the macro-aggregated accuracy.

At the genotype level, we simply count the number of correctly computed genotypes and divide this with the total number of genotypes:

$$\text{Genotype Acc.} = \frac{\sum_i(\#\text{ Correctly Imputed Individuals for Variant } i)}{\sum_i(\#\text{ of Individuals for Variant } i)}.$$

In the sensitivity vs positive predictive value (PPV) plots, the sensitivity and PPV are computed after filtering the imputed genotypes with respect to the imputation probability. We compute the sensitivity at the probability cutoff of τ is:

$$\text{Sens}_\tau = \frac{\sum_i(\#\text{ Correctly Imputed Individuals for Variant } i \text{ whose genotype probability} > \tau)}{\sum_i(\#\text{ of Individuals for Variant } i)}$$

Positive predictive value measures the fraction of correctly imputed genotypes among the genotypes whose probability is above the cutoff threshold:

$$\text{PPV}_\tau = \frac{\sum_i(\#\text{ Correctly Imputed Individuals for Variant } i \text{ whose genotype probability} > \tau)}{\sum_i(\#\text{ Individuals for Variant } i \text{ whose genotype probability} > \tau)}$$

Next, we swept a large cutoff range for τ from -5 to 5 with steps 0.01. We finally plotted the sensitivity versus PPV to generate the precision-recall curves for each method.

Micro-AUC accuracy statistics

For parameterizing the accuracy and demonstrating how different parameters affect algorithm performance, we used micro-AUC as the accuracy metric. This was also the original accuracy metric for measuring the algorithm performance in iDASH19 competition. Micro-AUC treats the imputation problem as a three-level classification problem where each variant is "classified" into one of three classes, i.e., genotypes, $\{0, 1, 2\}$. Micro-AUC computes an AUC metric for each genotype then microaggregates the AUCs for all the genotypes. This enables assigning one score to a multi-class classification problem. We use the implementation in scikit-learn package to measure the micro-AUC scores for each method (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html).

Measurement of time and memory requirements

For consistently measuring the time and memory usage among all the benchmarked methods, we used /usr/bin/time -f %e "\t" %M" to report the wall time (in seconds) and peak memory usage (in kilobytes) of each method.

Secure methods

We briefly describe the secure methods.

UTMSR-BFV and UTMSR-CKKS

The UTMSR (UTHealth-Microsoft Research) team uses a linear model with the nearby tag variants as features for each target variant. The plaintext model training is performed using the GNU Scientific Library. The collinear features are removed by performing the SVD and removing features with singular values smaller than 0.01. The target variant genotype is modeled as a continuous variable that represents the "soft" estimate of the genotype (or the estimated dosage of the alternate allele) and can take any value from negative to positive infinity. The genotype probabilities are assigned by converting the soft genotype estimation to a score in the range [0,1]:

$$p(g) = \exp(-1 \times |\tilde{g} - g|), g \in \{0, 1, 2\}, \qquad \text{(Equation 1)}$$

where g denotes one of the genotypes and $\tilde{g}$ represents the decrypted value of the imputed genotype estimate. Suppose that each variant genotype is modeled using genotypes of variants within k variant vicinity of the variant. In plaintext domain, the imputed value can be written as follows:

$$\tilde{g}_j = w_{j,0} + \sum_{r=1}^{k}\left(w_{j,r}^{-} \times g_{j-r}\right) + \sum_{r=1}^{k}\left(w_{j,r}^{+} \times g_{j+r}\right), \qquad \text{(Equation 2)}$$

where $w_{j,0}$ is the intercept of the linear model, and $w_{j,r}^{-}$ and $w_{j,r}^{+}$ denote the linear model weights for the j^{th} target variant's r^{th} upstream and downstream tag variants, respectively.

The secure outsourcing imputation protocols are implemented on two popular ring-based HE cryptosystems – BFV (Brakerski, 2012; Fan and Vercauteren, 2012) and CKKS (Cheon et al., 2017). These HE schemes share the same parameter setup and key-generation phase but have different algorithms for message encoding and homomorphic operations. In a nutshell, a ciphertext is generated by adding a random encryption of zero to an encoded plaintext, which makes the ring-based HE schemes secure under the

RLWE assumption. More precisely, each tag variant is first encoded as a polynomial with its coefficients, and the encoded plaintext is encrypted into a ciphertext using the underlying HE scheme. The plaintext polynomial in the BFV scheme is separated from an error polynomial (inserted for security), whereas the plaintext polynomial in the CKKS scheme embraces the error. Then Equation 2 is homomorphically evaluated on the encrypted genotype data by using the plain weight parameters. We exploit parallel computation on multiple individual data, and hence it enables us to obtain the predicted genotype estimates over different samples at a time. Our experimental results indicate that the linear model with 32 tag variants as features for each target variant shows the most balanced performance in terms of timing and imputation accuracy in the current testing dataset (see Table S8 and Figure S5). Our protocols achieve at least a 128-bit security level from the HE standardization workshop paper (Albrecht et al., 2018). We defer the complete details to the "UTHealth-Microsoft Research team solution" section in the supplementary document.

Chimera-TFHE

The Chimera team used multi-class logistic regression (logreg) models trained over one-hot encoded tag features: each tag SNP variant is mapped to 3 Boolean variables. Chimera's model training and architecture performed the best (with respect to accuracy and resource requirement) among six other solutions in the iDASH2019 Genotype Imputation Challenge.

We build three models per target SNP (one model per variant), i.e., target SNPs are also one-hot-encoded. These models give the probabilities for each target SNP variant. The maximal probability variant is the imputed target SNP value. A fixed number d of the nearest tag SNPs (in relation to the current target SNP) are used in model building. We train the models with different values of d in order to study the influence of neighborhood size: from 5 to 50 neighbors with an increment of 5. The most accurate model, in terms of micro-AUC score, is obtained for a neighborhood size $d = 45$. The fastest model with an acceptable accuracy (micro $-$ AUC$>$ 0.99) is obtained for $d = 10$. Although, the execution time of the fastest model is only ≈ 2 times faster compared to the most accurate model (refer to Table S2).

During the homomorphic evaluation, only the linear part of the logreg model is executed, which means in particular that we do not homomorphically apply the sigmoid function on the output scores. We use the coefficient packing strategy and pack as many plaintext values as possible in a single ciphertext. The maximum number of values that can be packed in a *RingLWE* ciphertext equals the used ring dimension, which is $n = 1024$ in our solution. We chose to pack one or several columns of the input (tag SNPs) into a single ciphertext. Since the TFHE library *RingLWE* ciphertexts encrypt polynomials with Torus ($\mathbb{T} = \mathbb{R}\text{mod}1$) coefficients, we downscale the data to Torus values (multiples of 2^{-14}) and upscale the model coefficients to integers.

In our solution, we use linear combinations with public integer coefficients. The evaluation is based on the security of *LWE* and only the encryption phase uses *RingLWE* security notions with no additional bootstrapping or key-switching keys. The security parameters have been tuned to support binary keys. Of course, as neither bootstrapping nor key-switching is used in our solution, the key distribution can be changed to any distribution (including the full domain distribution) without any time penalty. Our scheme achieves 130 bits of security, according to the LWE estimator (Albrecht et al., 2015). More information about the our solution is described in the supplementary document ("Chimera-TFHE team solution").

EPFL-CKKS

EPFL uses a multinomial logistic regression model with $d-1$ neighboring coefficients and 1 intercept variable for each target variant, with three classes {0,1,2}. The plaintext model is trained using the `scikit-learn` python library. The input variants are represented as values {0,1,2}. There is no pre-processing applied to the training data. For a target position j, the predicted probabilities for each class label are given by:

$$P\left[y = g \middle| z_r^{(p,j)}\right] = \frac{e^{w_0^{(\cdot,j,g)} + \sum_{r=1}^{d-1} w_r^{(\cdot,j,g)}}}{\sum_{i=0}^{2} e^{w_0^{(\cdot,j,i)} + \sum_{r=1}^{d-1} w_r^{(\cdot,j,i)} z_r^{(p,j,\cdot)}}}, \qquad \text{(Equation 3)}$$

where $\{w_0^{(\cdot,j,g)}, \ldots, w_{d-1}^{(\cdot,j,g)}\}$ are the trained regression coefficients for label $g \in \{0,1,2\}$ and position j, and $\{z_1^{(p,j,\cdot)}, \ldots, z_{d-1}^{(p,j,\cdot)}\}$ are the neighboring variants for patient p around target position j. The hard prediction for position j is given by $y^{(p,j,g)} = \text{argmax}_g\left(P\left[y = g \middle| z_r^{(p,j)}\right]\right)$. The variants $\{z_1^{(p,j,\cdot)}, \ldots, z_{d-1}^{(p,j,\cdot)}\}$ are sent encrypted and packed to the server, using the CKKS homomorphic cryptosystem, and the exponents in Equation 3 are computed homomorphically. The client decrypts the result and can obtain the label probabilities and hard predictions for each position. For the prediction, we use several numbers of regression coefficients, ranging from 8 to 64; as this number increases, both the obtained accuracy and the computational complexity increase (see Table S6). We use a single parametrization of the cryptosystem (see the "EPFL-Lattigo team solution" section in the supplemental information) for all the regression sizes, which keeps the cipher expansion asymptotically constant. The security of this solution is based on the hardness of the RLWE problem with Gaussian secrets.

SNU-CKKS

The SNU team applies one-hidden layer neural network for the genotype imputation. The model is obtained from Tensorflow module in plain (unencrypted) state, and the inference phase is progressed in encrypted stated for given test SNP data encrypted by the CKKS HE scheme. We encode each ternary SNP data into a 3-dimensional binary vector, i.e., $0 \rightarrow (1,0,0)$, $1 \rightarrow (0,1,0)$ and $2 \rightarrow (0,0,1)$. For better performance in terms of both accuracy and speed, we utilize an inherent property that each target SNP is mostly related by its adjacent tag SNPs. We set the number of the adjacent tag SNPs as a pre-determined parameter d, and run experiments on various choices of the parameter ($d = 8k$ for $1 \leq k \leq 9$). As a result, we check that $d = 40$ shows the best accuracy in terms of micro-AUC. Since the running time of computing genotype score grows linear to d, the fastest result is obtained at $d = 8$. We refer the intermediate value $d = 24$ to the most balanced choice in terms of accuracy and speed.

The security of the utilized CKKS scheme relies on the hardness of solving the RLWE problem with ternary (signed binary) secret. For the security estimation, we applied the LWE estimator (Albrecht et al., 2015), a sage module that computes the computational costs of state-of-art (R)LWE attack algorithms. The script for the security estimation is attached as a figure in the "SNU team solution" section in the supplementary document.

Non-secure methods

We describe the versions and the details of how the non-secure methods were run. The benchmarks were performed on a Linux workstation with 769 Gigabytes of main memory on an Intel Xeon Platinum 8168 CPU at 2.7 GHz with 96 cores. No other tools were run in the course of benchmarks.

Beagle

We obtained the jar formatted Java executable file for Beagle version 5.1 from the Beagle web site. The population panel (1,500 individuals) and the testing panel data are converted into VCF file format as required by Beagle. We ran Beagle using the chromosome 22 maps provided from the web site. The number of threads is specified as 16 threads at the command line (option 'nthreads=16'). We set the 'gp' and 'ap' flags in the command line to explicitly ask Beagle to save genotype probabilities that are used for building the sensitivity versus PPV curves. Beagle supplies the per genotype probabilities for each imputed variant. These probabilities were used in plotting the curves.

IMPUTE2

IMPUTE2 is downloaded from the IMPUTE2 website. The haplotype, legend, genotype, and the population panels are converted into specific formats that are required by IMPUTE2. We could not find a command line option to run IMPUTE2 with multiple threads. To be fair, we divided the sequenced portion of the chromosome 22 (from 16,000,000 to 51,000,000 base pairs) into 16 equally spaced regions of length 2.333 megabases. Next, we ran 16 different IMPUTE2 instances in parallel, as described in the IMPUTE2 manual. The output from the 16 runs is pooled to evaluate the imputation accuracy of IMPUTE2. IMPUTE2 provides per genotype probabilities, which were used for plotting the precision-recall curves.

Minimac3 and Minimac4

Minimac3 and Minimac4 are downloaded from the University of Michigan web site. We next downloaded Eagle 2.4.1 phasing software for phasing input genotypes. "Eagle+Minimac3" and "Eagle+Minimac4" were used in the Michigan Imputation Server's pipeline that is served for the public use. The panels are converted into indexed VCF files as required by Eagle, Minimac3, and Minimac4. We first used the Eagle protocol to phase the input genotypes. The phased genotypes are supplied to Minimac3 and Minimac4, and final imputations are performed. Eagle, Minimac3, and Minimac4 were run with 16 threads using the command line options ('–numThreads=16' and '–cpus 16' options for Eagle and Minimac3, respectively). Minimac3 and Minimac4 reports an estimated dosage of the alternate allele, which we converted to a score as in the above equation for UTMSR's scoring.

Minimac4 algorithm requires a preprocessing of the reference haplotype with a parameter estimation step. We observed that the parameter estimation step add a substantial amount of processing time and Minimac4 requires the parameter estimates to perform imputation.

Data and code availability

- Source data statement. Accuracy and resource benchmarking related source data have been deposited at https://doi.org/10.5281/zenodo.4947832. The 1000 Genomes project dataset are publicly available from NCBI portal at NCBI:ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/release/20130502/ALL.chr22.phase3_shapeit2_mvncall_integrated_v5a.20130502.genotypes.vcf.gz). The Illumino array platform metadata is available from https://support.illumina.com/downloads/human1m-duo_v3-0_product_
files.html.
- Code statement. The original source code, documentation, and usage examples for the imputation models are deposited at github: https://github.com/K-miran/secure-imputation and are also archived and deposited at https://doi.org/10.5281/zenodo.4948000.
- Scripts statement. The source and scripts for generating the figures and associated instructions are archived and deposited under https://doi.org/10.5281/zenodo.4947832 and are co-located with the figure-related datasets.
- Any additional information required to reproduce this work is available from the lead contact.

Cell Systems, Volume *12*

Supplemental information

Ultrafast homomorphic encryption models enable secure outsourcing of genotype imputation

Miran Kim, Arif Ozgun Harmanci, Jean-Philippe Bossuat, Sergiu Carpov, Jung Hee Cheon, Ilaria Chillotti, Wonhee Cho, David Froelicher, Nicolas Gama, Mariya Georgieva, Seungwan Hong, Jean-Pierre Hubaux, Duhyeong Kim, Kristin Lauter, Yiping Ma, Lucila Ohno-Machado, Heidi Sofia, Yongha Son, Yongsoo Song, Juan Troncoso-Pastoriza, and Xiaoqian Jiang

Supplementary Information for "Ultra-Fast Homomorphic Encryption Models enable Secure Outsourcing of Genotype Imputation"

Miran Kim[1,+], Arif Ozgun Harmanci[2,+,*,†], Jean-Philippe Bossuat[3], Sergiu Carpov[4,5], Jung Hee Cheon[6,7], Ilaria Chillotti[8], Wonhee Cho[6], David Froelicher[3], Nicolas Gama[4], Mariya Georgieva[4], Seungwan Hong[6], Jean-Pierre Hubaux[3], Duhyeong Kim[6], Kristin Lauter[9], Yiping Ma[10], Lucila Ohno-Machado[11], Heidi Sofia[12], Yongha Son[13], Yongsoo Song[14], Juan Troncoso-Pastoriza[3], and Xiaoqian Jiang[15, *]

[1]Department of Computer Science and Engineering and Graduate School of Artificial Intelligence, Ulsan National Institute of Science and Technology, Ulsan, 44919, Republic of Korea.
[2]Center for Precision Health, School of Biomedical Informatics, University of Texas Health Science Center, Houston, TX, 77030, USA.
[3]École polytechnique fédérale de Lausanne, Switzerland.
[4]Inpher, EPFL Innovation Park Bàtiment A, 3rd Fl, 1015 Lausanne, Switzerland.
[5]CEA, LIST, 91191 Gif-sur-Yvette Cedex, France.
[6]Department of Mathematical Sciences, Seoul National University, Seoul, 08826, Republic of Korea.
[7]Crypto Lab Inc., Seoul, 08826, Republic of Korea.
[8]Zama, Paris, France and imec-COSIC, KU Leuven, Leuven, Belgium.
[9]Microsoft Research, Redmond, WA, 98052, USA.
[10]University of Pennsylvania, Philadelphia, PA, 19104, USA.
[11]UCSD Health Department of Biomedical Informatics, University of California, San Diego, CA, 92093, USA.
[12]National Institutes of Health (NIH) - National Human Genome Research Institute, Bethesda, MD, 20892, USA.
[13]Samsung SDS, Seoul, Republic of Korea.
[14]Department of Computer Science and Engineering, Seoul National University, Seoul, 08826, Republic of Korea.
[15]Center for Secure Artificial intelligence For hEalthcare (SAFE), School of Biomedical Informatics, University of Texas Health Science Center, Houston, TX, 77030, USA.
[†]Lead Contact Author: Arif.O.Harmanci@uth.tmc.edu
[*]Corresponding authors: Arif.O.Harmanci@uth.tmc.edu, Xiaoqian.Jiang@uth.tmc.edu
[+]These authors contributed equally to this work

Description of the iDASH19 Challenge Setup and Results

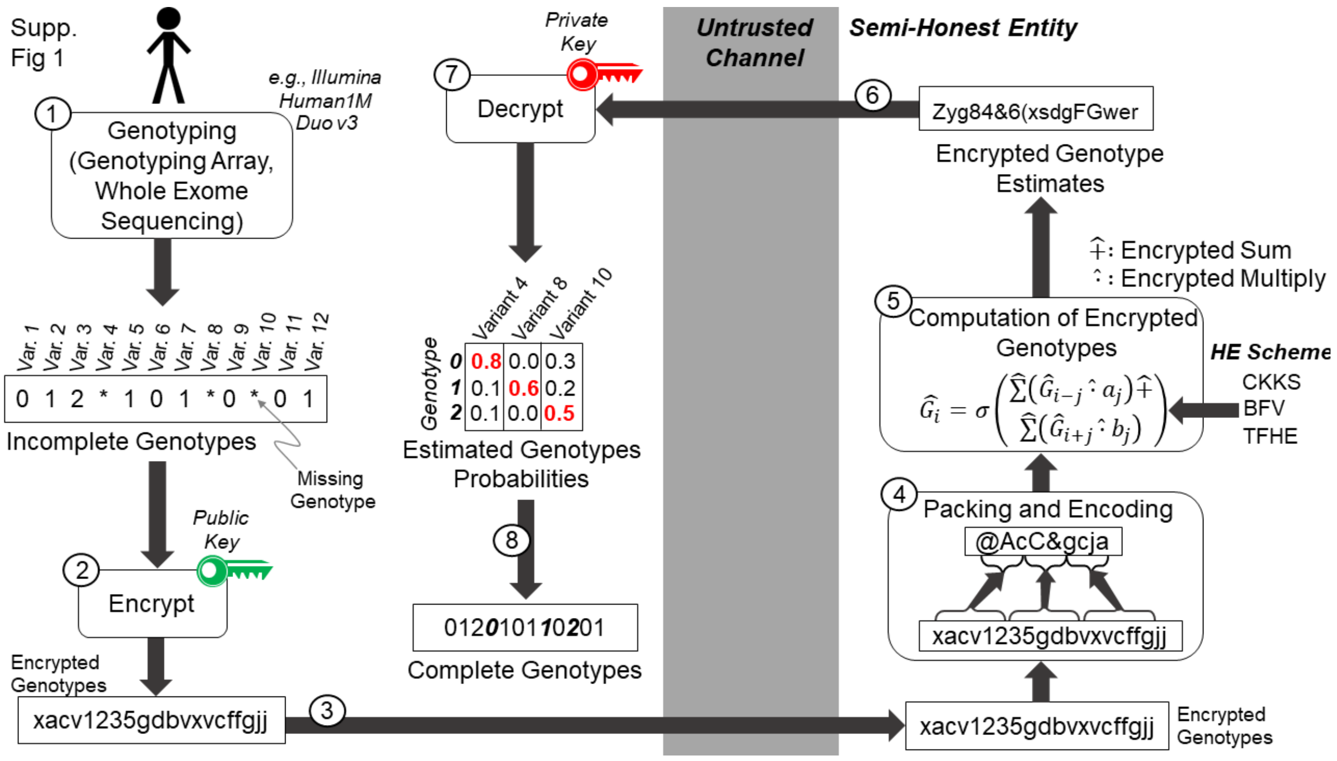
Our study is inspired by the 2019 iDASH Genomic Privacy Challenge for the development of secure tools to enable genotype imputation (Fig. 1). Before the challenge, the organizers put together a training dataset with the tag and target variants, which served as the challenge dataset. The dataset was later distributed to all the participants, who were tasked with building the imputation models and implementing secure versions of them. Each team submitted their models as docker images to the organizers before the closing of the challenge. There were more than 49 registrations from all around the globe and the organizers received 6 final submissions with solutions. The organizers then benchmarked all the submitted methods on a separate dataset that had been withheld from the participants. Based on the encouraging results from the participants, the top 3 performing teams (Chimera, EPFL, SNU) and the organizer team (UTHealth and Microsoft Research) collaborated to do large scale testing and development of the imputation methods.

In the iDASH competition, we aimed at testing the feasibility of developing secure imputation models. To ensure that the task was solvable in a limited amount of time, we focused on two regions on chromosome 1 (chr1:15812664-25811728 and chr1:186118607-196115993) with medium heterozygosity and gene density (for exclusion of regions under high selection pressure), where two training datasets were developed. The first dataset contained 9,746 tag Single Nucleotide Polymorphisms (SNPs) and the second contained 1,045 tag SNPs. Both datasets were used to predict genotypes of 500 target SNPs. Since both variant sets span the same region, the first dataset had lower variant-variant distance (approximately 1900 base pairs), i.e., higher density of tag variants compared to the second region (approximately 18,000 base pairs). Hence, the first dataset, denoted by 1K (name referring to the average distance between consecutive tag variants), represents an easier imputation task compared to the second one, denoted by 10K. The results of the challenge are summarized in Supplementary Table 1.

Supplementary Table 1. The micro-AUC and timing benchmarks for the methods compared in the iDASH19 Genomic Privacy Homomorphic Encryption Challenge.

Team	Dataset	Model	Accuracy	Memory	Time (seconds)			
				(GB)	Encryption	Evaluation	Decryption	Total
Chimera-TFHE	1K	$1K_a$	0.9971	0.1542	2.6761	0.9583	0.2062	3.8406
		$1K_b$	**0.9971**	**0.1500**	2.6441	0.7063	0.1991	**3.5496**
		$1K_c$	0.9971	0.1541	2.6167	0.9702	0.1992	3.7860
	10K	$10K_a$	0.9763	0.0339	0.3047	0.2310	0.1978	0.7334
		$10K_b$	**0.9763**	**0.0363**	0.3099	0.3146	0.1933	**0.8179**
		$10K_c$	0.9759	0.0382	0.3063	0.4202	0.1971	0.9236
SNU	1K	$1K_a$	**0.9966**	**0.9023**	53.9480	102.6470	2.0912	**158.6862**
	10K	$10K_a$	0.9729	0.5943	5.7560	107.8530	1.8613	115.4703
		$10K_b$	**0.9750**	**0.6088**	5.7150	43.6859	1.9330	**51.3339**
EPFL	1K	$1K_a$	0.9930	0.6614	7.2200	2.3500	0.6400	10.1870
		$1K_b$	**0.9936**	**4.3152**	13.0700	8.2900	0.8600	**21.8590**
		$1K_c$	0.9932	1.6825	8.3700	4.8900	0.4600	13.7600
	10K	$10K_a$	0.9682	1.7017	7.8100	2.2300	0.5100	10.5840
		$10K_b$	**0.9705**	**4.2926**	13.1300	8.0200	0.7900	**22.8740**
		$10K_c$	0.9688	1.7017	8.4700	4.9500	0.4700	13.9200

section*Genotype Imputation Problem and Security Guarantees We describe the main focus of the genotype imputation problem and the security guarantees provided by the secure schemes. *Focus on the Unphased Genotype Imputation.* In the genotype imputation scenario, it is important to highlight the fact that we are solving the "unphased genotype imputation problem". This is important to highlight because the current state-of-the-art non-secure methods (such as Minimac3, Minimac4[1]) impute the variants when the phased genotypes are available, which means that the genotypes must be phased into maternal



Supp. Fig 1
1
Genotyping (Genotyping Array, Whole Exome Sequencing)
e.g., Illumina Human1M Duo v3
Var. 1 Var. 2 Var. 3 Var. 4 Var. 5 Var. 6 Var. 7 Var. 8 Var. 9 Var. 10 Var. 11 Var. 12
0 1 2 * 1 0 1 * 0 * 0 1
Incomplete Genotypes
Missing Genotype
2
Public Key
Encrypt
Encrypted Genotypes
xacv1235gdbvxvcffgjj
3
7
Private Key
Decrypt
Variant 4 Variant 8 Variant 10
Genotype
0 0.8 0.0 0.3
1 0.1 0.6 0.2
2 0.1 0.0 0.5
Estimated Genotypes Probabilities
8
012010110201
Complete Genotypes
Untrusted Channel
Semi-Honest Entity
6
Zyg84&6(xsdgFGwer
Encrypted Genotype Estimates
$\hat{+}$: Encrypted Sum
$\hat{\cdot}$: Encrypted Multiply
5
Computation of Encrypted Genotypes
$\hat{G}_i = \sigma\left(\begin{matrix}\hat{\sum}(\hat{G}_{i-j} \hat{\cdot}\, a_j)\hat{+} \\ \hat{\sum}(\hat{G}_{i+j} \hat{\cdot}\, b_j)\end{matrix}\right)$
HE Scheme
CKKS
BFV
TFHE
4
Packing and Encoding
@AcC&gcja
xacv1235gdbvxvcffgjj
xacv1235gdbvxvcffgjj
Encrypted Genotypes

Supplementary Figure 1. Overview of the secure imputation scenario.

and paternal chromosomes using, for example, EAGLE[2] before they can be imputed using, for example, Minimac3. From the privacy and security perspective, the phasing step must be also protected and requires high computational resources to accomplish. To provide a complete privacy framework and security of the genotype data, we aim at imputing the target variant genotypes directly from the unphased genotypes so that there is no need for a separate phasing step where genotypes are sent to another untrusted semi-honest entity. It is worth noting that our approaches currently do not handle the phased genotype imputation problems but they can be adapted for phased genotype imputation after proper re-training of the models for this problem. This requires, however, changes in model architectures and data representation and we view this as a separate task from our current focus in this study.

As we describe in detail below, we developed a number of secure imputation methods. In this study, we focus primarily on linear imputation models that can be trained very fast, using optimized libraries [3]. One of the key points of secure and privacy-preserving imputation is that the parameters of the imputation models can be stored openly (or in “plaintext”) without the need of protection as they depend only on the population panels that are publicly available. Thus, the imputation server can train the imputation models without requiring secure computations. Using the optimized scientific libraries, the linear models can be trained very efficiently. This way, the imputation can be performed at near real-time training potential in the case that the model parameters need to be trained when a new set of tag variants are used in the imputation. This is necessary when a user provides a custom-built array platform [4] that the server did not process previously and the imputation models are trained from scratch. To utilize the trained imputation models for each SNP in the course of imputation, the arithmetic operations of the imputation model are securely and efficiently computed, where one of the operands (the genotype) is encrypted and the model parameters are plaintext (Fig. 1b, 1c). We briefly describe our methodological approaches.

Imputation Security Guarantees. We focus on three different types of homomorphic encryption "schemes", namely the BFV [5,6], CKKS [7], and TFHE [8] schemes, and we provide implementations of the imputation models that use these schemes. These schemes use different variants of a basic principle named "lattice-based hardness assumption" to provide security guarantees while the genotype imputation is performed. In more formal terms, these schemes provide indistinguishability of the encrypted data under "chosen plaintext attacks" (termed IND-CPA condition[9]), which is a standard security notion that guarantees confidentiality of the data. It should be noted that there are many more approaches and guarantees for security and confidentiality of the data (such as authenticated encryption, which provides confidentiality and integrity of ciphertexts) but these may be incompatible with HE-based frameworks. These formalisms should be considered in future studies. The CPA attacks represent scenarios where the attacker (or a cryptanalyst) can use the publicly available key to encrypt carefully selected messages to generate encrypted data and analyze the characteristics of the encryption to reveal information about the encryption schemes, such as non-uniformity of encrypted messages, which can leak information about sensitive data. Previous research has shown that the lattice-based systems guarantee the indistinguishability (and security) condition because it is practically infeasible for a computationally bounded adversary to decrypt encrypted data without the access to the private key. The different schemes provide different interfaces and differences in details of the implementation of this basic principle.

Alternative Security Approach by “Model Sharing”. In the imputation scenarios considered here, it should be noted that the variant positions are passed between the server and the researcher in plaintext format. This can potentially lead to information leakage. For instance, the number of target variants can leak the information about the technology that was used for genotyping and potentially reveal the identity of the researchers or the research institution. These types of attacks are very subtle and highlight the need for formal approaches such as HE to encrypt all the data. One practical way to ensure these leakages are patched is to ensure that the tag and target variants are standardized. Usage of standard array platforms is one way to do this: The tag variants are standardized and the targets are defined by the publicly available reference panels. This is also useful for non-secure imputation methods since imputation across genotyping platforms was shown to induce biases because of the difference in the used tag SNPs[10]. The authors argue that the best strategy for mitigation of these biases is to use the common tag variants across arrays, which is similar to standardization of the used tag variants.

Similarly, one approach that a client can take is to download the models that are trained by the imputation server and evaluate them locally for imputing variants. This idea resembles "model sharing" paradigms for protecting privacy[11]. This is also useful since the models have a much smaller footprint compared to the reference panels. Especially when the imputation is performed on a small region and a few genomes, downloading the imputation models and local plaintext evaluation approach can prove practical. This, however, requires the server to make the newly trained models available for download. To follow up the local imputation scenario, it should be noted that the implemented approaches are platform independent and do not require a specific hardware platform, unlike other approaches such as Intel’s Secure Guard Extensions (SGX) or AMD’s Secure Encrypted Virtualization (SEV), which require specific hardware on both the server and client sides. These approaches provide a security-by-engineering (with potential side-channel and hypervisor vulnerabilities) approach unlike HE and MPC approaches that provide theoretical security of the data.

Comparison of the Secure Methods

To compare the performance of the proposed models, we first analyzed how the accuracy changes over the genome. Supplementary Figure 8a shows the distribution of accuracy over chromosome 22 coordinates. We observed that the methods show similar trends of accuracy over the chromosome. We also observed that there are certain spans of consecutive variants that are harder to impute consistently among different methods. Evaluating the accuracy among different methods, UTMSR-LMSE's ordinary linear method shows a wide range of accuracy values compared to other methods. This is more clear when we evaluate the MAF vs the accuracy of different methods (Supplementary Figure 8b). From this, the ordinary linear method shows more outliers which highlights the sporadically less accurate performance of the ordinary linear model. Interestingly EPFL's logistic regression method (one of the best overall methods) exhibited a patch of variants with very low MAF that were of low accuracy. These results highlight the fact that trained models show a diverse range of accuracies. We next went on to perform pairwise comparisons of methods (Supplementary Figure 8c). UTMSR-LMSE's wide range of accuracies are better visualized in this figure (bottom row in Supp. Figure 8c). Overall, EPFL's logistic regression imputation performs favorably in comparison to the other methods.

In summary, each method shows a variety of accuracy values depending on MAF and genomic location. We, therefore, would like to point out that this recommendation should be taken mildly as these methods can be further improved and fine-tuned by tag SNP selection[12] and methods such as ensembling. For instance, we observed that the average accuracy increases around 0.5% (95.3% to 95.8%) when we computed accuracy using the best imputer at each variant. While this is an overly optimistic estimation of the accuracy, it suggests that combining different approaches can provide higher accuracy. Also, we observed that logistic regression-based methods can be trained to perform better than other methods that our teams trained for the variants with low allele frequencies (Figure 2k). We believe our results suggest it is necessary to improve these models to observe some of the expected gains in accuracy. This is one of the current major limitations of our approach to imputation and should be further investigated in the future.

When we compare the methods in terms of their resource usages, the ordinary linear model is, as expected, required the least amount of time and memory. It is interesting, however, that the resource requirements of the logistic regression methods (Chimera and EPFL) and the simple neural networks method (SNU) are very feasible for secure evaluation purposes. We foresee that these methods can be engineered to provide feasible performance for real-time training and evaluation for genotype imputation purposes. In fact, our teams have not made use of numerous optimizations such as clustering of the nearby target variants to increase the imputation speed and optimizing the model complexity at sites with lower recombination rates. The HMM-based methods naturally optimize performance in these regions by techniques such as state-collapsing[13].

Discussions about Limitations of HE-based Imputation Methods

We briefly describe and discuss the limitations of the HE-based imputation methods.

Firstly, there are inherent limitations in Homomorphic Encryption schemes that must be overcome by the theoreticians. For instance, HE schemes are still limited practically by the multiplicative depth of an evaluation circuit. We are, however, very hopeful that a lot of these challenges can be overcome by new schemes and approaches. Our motivation for this is the fact that HE schemes have made seen big progress in the last decade, more than 5 orders of magnitude improvement has been achieved since the inception of the HE computations. These limitations can partially be overcome by re-engineering and re-factoring the imputation algorithms but this again requires much in-depth analysis of the algorithms and cross-disciplinary expertise in genomics and cryptography. The second limitation stems from the complexity of homomorphic encryption algorithms. Expertise in these fields requires substantial training in the fields of computer science and mathematics. This makes it necessary to put a large interdisciplinary team together to develop new imputation algorithms.

Third, unlike the HMM-based methods that rely on the reference panels, the training of our vicinity-based methods needs to be performed on the fly. This can be particularly compute-intensive for methods such as neural networks. Thus, our results on resource requirements can be overly optimistic. We, however, have observed that the logistic regression currently performs fairly well and plaintext-training of these models can be performed fairly fast. These methods can enable a good performance accuracy tradeoff in training and secure evaluation.

Approaches of the Imputation Methods and Cryptographic Innovations

UTHealth-MSR.

Innovations in Plaintext training. We have developed the ordinary least squares model training using GNU Scientific Libraries (GSL) in C++ to optimize the memory/time usage. We used the singular value decomposition (SVD) to remove the collinearities of the tag variant genotypes before training of the ordinary least squares (OLS) models.

Innovations in Cryptography. We proposed a general and unified solution to evaluate a linear prediction model for genotype imputation, which can be applied to two popular HE cryptosystems – BFV and CKKS. We exploit the single instruction multiple data (SIMD) representation in which each variant of the predictor tag genotypes is encoded as a polynomial with its coefficients and encrypted using the underlying HE scheme. Based on our optimized encryption method, homomorphic computation is performed over encrypted genotype data by simple constant-ciphertext multiplications using the plain real-valued regression coefficients of the linear prediction models.

Chimera.

Innovations in Plaintext training. We trained all the models for our experiments by using state-of-the-art logistic regression on one-hot encoded features, i.e., by mapping each tag and target SNP to 3 independent scores. The trainings have been done by using the *Vowpal Wabbit*[14] machine learning system.

Innovations in Cryptography. In order to perform the homomorphic evaluation, we interpreted the trained models as plain linear regression (instead of logistic regression) models. This trick allowed us to avoid the evaluation of the sigmoid function on the output scores, by keeping an excellent accuracy in the predictions. We also took advantage of the polynomial packing of the RingLWE ciphertexts in the TFHE scheme in order pack multiple SNP in a single ciphertext, resulting in an improvement of the homomorphic evaluation, both in terms of execution time and ciphertext sizes. We encoded the cleartext trained model in a sparse matrix: the sparsity is due to the fact that most of the scores only depend on a small number of neighbours SNPs. Thanks to all of this improvements, the final homomorphic imputation is performed by simply doing constant-ciphertexts multiplications and leveled additions.

EPFL.

Innovations in Plaintext training. We relied on a standard baseline approach for the training process, in order to show that simple logistic regression models that can be efficiently evaluated with homomorphic encryption already provide an accuracy that is close to non-secure state-of-the-art solutions.

Innovations in Cryptography. Based on the prior knowledge of which computations have to be performed for prediction, we optimized the encoding of the to-be-encrypted values and we achieved a $2\times$ gain on the number of values that can be packed in a single ciphertext. We also halved the amount of data that has to be sent from the client to the server by only sending "seeds" of polynomials to the server, which can deterministically generate the corresponding complete polynomials.

SNU.

Innovations in Plaintext training. We trained a neural network with one hidden-layer using public python machine learning library *keras*. We experimentally set the best choices on the number of nodes (64) in the hidden layer, and use a linear activation function for hidden layer and Sigmoid for output layer. A linear activation function derives better accuracy than the others such as ReLU and Sigmoid. In addition, each SNP feature is one-hot encoded as a vector to obtain an independent score for each feature, which also shows better accuracy comparing the model without the one-hot encoding.

Innovations in Cryptography. Since our model is trained with a linear activation function for the hidden layer, it is evaluated by a simple matrix multiplication followed by Sigmoid function. For homomorphic evaluation, we omit the last Sigmoid evaluation by observing that the order of scores is unchanged by Sigmoid. We also use a trick that halves the number of ciphertexts by encoding two real numbers into one complex number, and using CKKS scheme that supports encryption of complex numbers.

Description of the Security Guarantees for Homomorphic Encryption Systems

After the first plausible construction of HE[15], this family of cryptosystems has progressed significantly towards both basing the security on more standard assumptions and improving their efficiency. Currently, the ring-based HE schemes, such as BFV[5,6], CKKS[7], and TFHE[8], have shown good performance on real-world problems. Throughout the paper, we assume that n is a power-of-two integer and $R = \mathbb{Z}[X]/(X^n + 1)$. We write $R_q = R/(q \cdot R)$ for the residue ring of R modulo an integer q. The Ring Learning with Errors (RLWE) assumption with parameters (n, q, χ, ψ) states that given any polynomial number of samples of the form $(a_i, b_i = s \cdot a_i + e_i) \in R_q^2$, where a_i is uniformly random in R_q, s is chosen from the key distribution χ over R_q, and e_i is drawn from the error distribution ψ over R, the b_i's are computationally indistinguishable from uniformly random elements of R_q. Then, the ring-based HE schemes are IND-CPA secure if the RLWE problem of secret s with parameter (n, q, χ, ψ) is hard.

Chimera-TFHE Team Solution

In this section, we describe the solution of the Chimera-TFHE team (ranked 1st at the iDASH 2019 competition). In this solution, we use the fully homomorphic encryption library TFHE[8] and its Chimera leveled homomorphic encryption variant[16] (more precisely, we apply approximate number encoding over the torus).

Method overview

All the models are trained in plain by performing a multi-class *logistic regression* (logreg) over one-hot encoded features: each tag SNP variant is mapped to three Boolean variables (either if SNP is homozygous, heterozygous or homozygous alternate). We build three models per target SNP (one model per variant), i.e., target SNPs are also one-hot-encoded. A fixed number d of the nearest tag SNPs (in relation to the current target SNP) are used in model building. To study the influence of neighborhood size, we train the models with different values of d: from 5 to 50 neighbors with a 5 increment. The number of regression coefficients of a model is equal to $3 \times d$ plus one coefficient for the intercept part. In this way, the size of obtained models is under control and over-fitting is limited. Like in standard logistic regression multi-class predictor, the output class between the three target SNP variant models is the one having the maximal score. The Vowpal Wabbit[1] machine learning system with default configuration parameters is used for training the logreg models.

During the homomorphic evaluation only the linear part of the logreg model is executed, which means in particular that we do not homomorphically apply the sigmoid function on the output scores. The scores produced by our implementation can be positive or negative and are used directly in the evaluation of method performance.

Plaintext Version We start by describing the plaintext version of the imputation algorithm followed by the actual implementation of this algorithm on encrypted data. We denote by $3 \cdot L$ the number of input features (one-hot encoded tag SNP values of L tag SNPs), by M the number of samples (individuals) and by $3 \cdot T$ the number of target SNPs to predict (one-hot-encoded target SNP values for T target SNPs). Indeed, each tag or target SNP has 3 possible values (0, 1 or 2), which are treated as independent features.

The inputs to the algorithm are

- $X = (X[i][j])_{i \in [M], j \in [3L]}$ – contains the input features as Boolean values, i.e. the one-hot-encoded tag SNPs for each individual.

- $W = (W[j][k])_{j \in [3L], k \in [3T]}$ – contains the coefficients of the model, given as a sparse matrix because each target SNP model depends only on a small fixed number d of neighbour tag SNPs.

The evaluation of all the models consists in a multiplication between these two matrices, that is to say $Y = X \cdot W$ where $Y = (Y[i][k])_{i \in [M], k \in [3T]}$ is the matrix of scores for each target SNP variant. Naturally, the sparseness of the matrix W is taken into account in the actual implementation.

Ciphertext Version In the encrypted evaluation of the regression models, the coefficient matrix W is public and the matrix of tag SNP variants X is encrypted. RingLWE homomorphic encryption schemes enable us to pack many plaintext messages into a single ciphertext using different packing strategies (e.g., slot or coefficient). We use the coefficient packing strategy and pack as many plaintext values as possible into a single ciphertext. The maximum number of values that can be packed into a RingLWE ciphertext equals the used ring dimension, which is $n = 1024$ in our solution. We chose to pack one or several columns of matrix X into a single ciphertext (per tag SNP packing). The exact number of columns we pack is $B = \lfloor n/M \rfloor$, which is the maximal integral number of matrix X columns that can be packed into a ciphertext.

Since the TFHE library RingLWE ciphertexts encrypt polynomials with Torus ($\mathbb{T} = \mathbb{R} \mod 1$) coefficients, we downscale the data X to Torus values (multiples of 2^{-14}) and upscale the model coefficients W to integers. The downscale and upscale ratios are chosen so that the plaintext evaluation $Y = X \cdot W$ on training data does not overflow over the Torus (our model outputs are bounded to a 1/2 range to leave a 100% safety margin).

In the iDASH 2019 competition dataset, since the number of lines of the input matrix X is $M = 500$, we extended our encoding to pack $B = 2$ entire columns of X into a single RingLWE ciphertext: The first column is typically placed in the first 500 coefficients of the encrypted polynomial, while the second column is encoded in the next 500 coefficients. The remaining coefficients in the polynomial are unused (in practice, set to 0).

The pseudo-code of the imputation algorithm on encrypted data is given in Algorithm 1. The input matrix X coefficient encrypted in a RingLWE ciphertext are $X[.][B \cdot j, B \cdot j + 1, \ldots, B \cdot (j+1) - 1]$ for $j \in [3L/B]$. The regression coefficients W are given in clear form. The prediction scores are also RingLWE ciphertexts with the first M coefficients set to individual's scores and the others random. The algorithm is straightforward, encrypted input features are multiplied by plaintext model coefficients. Temporary ciphertexts (line 2) are used to manage B columns of the matrix X independently. Each such ciphertext contains a part of the final score. Rotated versions of these ciphertexts are added together to obtain final scores (line 10). Finally, we randomize coefficients of output ciphertext (line 11) to avoid information leakage other than regression scores. Each of the obtained $3T$ RingLWE$(Y[.][k])$ ciphertexts contains the prediction for one target SNP variant for all individuals.

[1] `https://github.com/VowpalWabbit/vowpal_wabbit`, version 8.5.0

Algorithm 1 Homomorphic Evaluation.

```
Require: Input features $\mathsf{RingLWE}(X[.][B\cdot j,\dots,B\cdot(j+1)-1])$ for $j\in[3L/B]$    ▷ For simplicity we suppose that $B|3\cdot L$
Require: Regression coefficients $W[j][k]$ in clear for $j\in[3L]$ and $k\in[3T]$
Ensure: Prediction scores $\mathsf{RingLWE}(Y[.][k])$ for $k\in[3T]$

1:  for $k=0,\dots,3T-1$ do
2:      $Y_0[.]=0, Y_1[.]=0,\dots,Y_{B-1}[.]=0$    ▷ Initialize $B$ temporary values
3:      for $j=0,\dots,3L/B-1$ do
4:          for $l=0,\dots,B-1$ do
5:              if $W[j+l][k]\neq 0$ then
6:                  $\mathsf{RingLWE}(Y_l[.]) += W[j+l][k]\cdot\mathsf{RingLWE}(X[.][B\cdot j,\dots,B\cdot(j+1)-1])$
7:              end if
8:          end for
9:      end for
10:     $\mathsf{RingLWE}(Y[.][k])=\mathsf{RingLWE}(Y_0[.])+\mathsf{RingLWE}(Y_1[.])\cdot X^{-M}+\dots+\mathsf{RingLWE}(Y_{B-1}[.])\cdot X^{-(B-1)\cdot M}$
11:     $\mathsf{RingLWE}(Y[M,\dots,n][k])=0$    ▷ Randomize unused coefficients
12: end for
```

Parameters and Security

The practical parameters we chose to instantiate in the RingLWE encryption are $n=1024$, the ring dimension and the size of the binary secret key, and $\alpha=2^{-25}$, the standard deviation used to sample the initial Gaussian noise in the ciphertexts. Our scheme achieves 130 bits of security, according to the LWE estimator[17].

Experimental Results

Learning plaintext logistic regression models (i.e. matrix W coefficients) was performed on a machine with an Intel Xeon CPU @ 2.0GHz processor, with 64GB of RAM, using 8 execution threads. Learning time is between 7 and 10 hours, from smallest to largest vicinity size (tag SNPs neighbourhood size). The number of obtained logreg models is more than 240k ($=3\cdot T$) for each vicinity value. The disk space for archived models is between 24MB and 158MB.

We have homomorphically executed the logreg models for different number of target SNPs (20k, 40k and 80k), different number of tag SNP neighbourhood sizes and population stratification (AFR, AMR and EUR). The tests were performed on a machine with an Intel Xeon Platinum 8168 Processor @ 2.7GHz, 64G RAM using 16 threads. The training and evaluation phases code is publicly available on github https://github.com/ssmiler/idash2019_2.

Table 2 presents execution metrics[2] as a function of number of target SNPs (column "#target SNPs") and vicinity size (column "#closest tag SNPs"). The HE key generation time (column "Keygen") is a one-time process and independent on the encrypted data or the executed computation. This column contains three values only for aesthetic reasons. The encryption and decryption times are independent of the regression model (same value in columns "Encr." and "Decr.") and depend only on the number of target SNPs. We chose to depict only the total RAM memory used during the evaluation phase (column "Evaluation RAM (GB)") because it is the largest one of the four steps. In the worst case, the evaluation phase uses less than 6GB of RAM. The sizes of encrypted data (tag SNP variants) and result (target scores) is given in the last two columns. In the context of cloud HE-based imputation, column "Input data size (MB)" represents the size of data to upload to the cloud and column "Score data size (MB)" the size of result to download from the cloud. As expected, the evaluation time (column "Eval.") increases with the vicinity size in a linear manner.

In table 3 are shown accuracy metrics as a function of vicinity size and population stratification. Table lines (labeled "All", "AFR", "AMR" and "EUR") represent the stratification of the population. The metrics were computed on the test partition of the input data. We can observe logreg model over-fitting after a certain vicinity size, e.g., models obtain worse accuracy metrics for a vicinity larger that 25 for "AMR" and "EUR" stratification.

Using the values in tables 2 and 3 one can find the best compromise between model accuracy and evaluation time.

Table 4 illustrates the acceleration capability of our solution for different number of execution cores. Logistic regression imputation model with a vicinity of 10 tag SNPs is used.

[2]Serialization time is not included in the illustrated timings. We note that the serialization takes about 60-80 % of the total execution time.

Supplementary Table 2. Experimental results of HE-based genotype imputation – Team TFHE-Chimera.

#target SNPs	#closest tag SNPs	Time (sec.) Keygen	Encr.	Eval.	Decr.	Total	Evaluation RAM (GB)	Input data size (MB)	Score data size (MB)
20k	5	3.90E-05	0.169	0.435	0.182	0.785	0.65	81.7	469.67
	10			0.563		0.913	0.72		
	15			0.772		1.122	0.81		
	20			0.873		1.223	0.87		
	25			0.983		1.334	0.92		
	30			1.175		1.526	1.05		
	35			1.315		1.665	1.11		
	40			1.462		1.812	1.16		
	45			1.611		1.961	1.22		
	50			1.718		2.068	1.27		
40k	5	3.78E-05	0.355	0.881	0.369	1.604	1.32	187.91	939.33
	10			1.202		1.925	1.46		
	15			1.631		2.354	1.65		
	20			1.917		2.640	1.76		
	25			2.104		2.828	1.87		
	30			2.823		3.546	2.13		
	35			2.822		3.546	2.24		
	40			3.371		4.095	2.35		
	45			4.072		4.796	2.46		
	50			4.125		4.848	2.57		
80k	5	3.86E-05	0.716	1.990	1.078	3.784	2.67	378.79	1899.37
	10			2.481		4.275	2.96		
	15			3.583		5.377	3.34		
	20			4.165		5.959	3.56		
	25			4.643		6.437	3.77		
	30			5.680		7.474	4.32		
	35			6.896		8.690	4.53		
	40			7.176		8.970	4.75		
	45			8.152		9.946	4.97		
	50			8.910		10.704	5.18		

Supplementary Table 3. Accuracy metrics – Team TFHE-Chimera.

#closest tag SNPs		5	10	15	20	25	30	35	40	45	50
All	uAUC	0.9815	0.9901	0.9923	0.9932	0.9936	0.9938	0.9939	0.9940	**0.9940**	0.9940
	MAP	0.8956	0.9303	0.9415	0.9462	0.9485	0.9498	0.9505	0.9509	0.9510	**0.9511**
	Non-ref MAP	0.7290	0.8277	0.8600	0.8734	0.8802	0.8840	0.8862	0.8876	0.8882	**0.8886**
AFR	uAUC	0.9712	0.9816	0.9846	0.9857	0.9862	**0.9863**	0.9862	0.9861	0.9859	0.9856
	Non-ref MAP	0.6706	0.7610	0.7943	0.8084	0.8148	0.8175	**0.8178**	0.8175	0.8163	0.8147
AMR	uAUC	0.9865	0.9917	0.9928	0.9931	**0.9931**	0.9930	0.9929	0.9927	0.9925	0.9923
	Non-ref MAP	0.7551	0.8337	0.8536	0.8579	**0.8586**	0.8572	0.8543	0.8513	0.8481	0.8447
EUR	uAUC	0.9905	0.9948	0.9956	0.9959	**0.9959**	0.9959	0.9958	0.9957	0.9956	0.9954
	Non-ref MAP	0.7765	0.8485	0.8655	0.8698	**0.8702**	0.8686	0.8673	0.8649	0.8629	0.8607

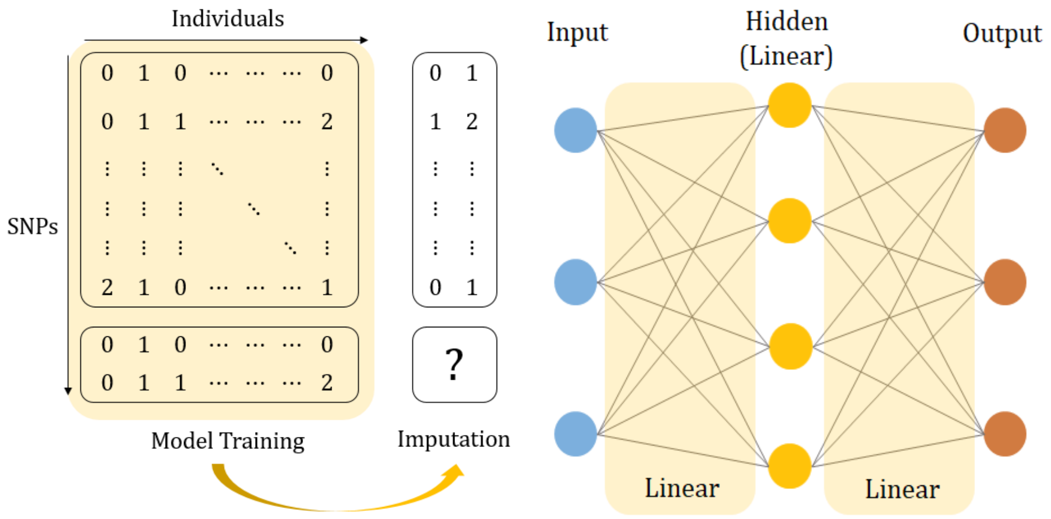
Supplementary Table 4. Total running time (in seconds) as a function of used core count – Team TFHE-Chimera.

# target SNPs / # cores	1	2	4	8	16	24
20K	4.637	2.740	1.753	1.204	0.913	0.816
40K	11.001	6.262	3.714	2.492	1.925	1.696
80K	21.468	14.181	8.888	5.884	4.275	3.645

SNU Team Solution

In this section, we describe the solution submitted by the Seoul National University team (ranked 2nd place at iDASH19 Genome Privacy Challenge). We adopted one-hidden layer neural network for the genotype imputation (See Supplementary Figure 2), and our solution is based on the HEaaN library, which is an implementation of the CKKS[7] scheme. Our one-hidden layer with linear activation setting provides us, according to our experiments, more accurate results than a simple logistic regression setting. Also, we set the activation function in the hidden layer as the linear function, not ReLU or Sigmoid, because the linear function gave us higher accuracy than others. Since we use a single hidden layer with linear activation, the output model after the training phase consists of two weight matrices $W^{(1)}$ and $W^{(2)}$, and we can pre-compute the multiplication of these matrices $W = W^{(1)} \cdot W^{(2)}$ before the evaluation phase. In other words, in the evaluation phase, we only need to multiply one pre-computed weight matrix for each model.

We encoded all SNP data $0,1,2$ by $(1,0,0)$, $(0,1,0)$ and $(0,0,1)$, respectively, so the dimension of SNP data is three times increased. We denote by $X_1 \in \{0,1\}^{M_0 \times 3L}$ and $Y_1 \in \{0,1\}^{M_0 \times 3T}$ the encoded training tag SNP and target SNP matrices, respectively, where M_0 is the number of individuals in training data. We similarly define the encoded test tag/target SNP matrices as $X_2 \in \{0,1\}^{M \times 3L}$ and $Y_2 \in \{0,1\}^{M \times 3T}$.



Supplementary Figure 2. We train a model with top two rectangles(in left figure), then predicts emptied bottom right rectangle with bottom left rectangle data. Our model consists of one hidden linear layer (in right figure).

Plain Model Generation

The most naive approach is to simply train a neural network model from (X_1, Y_1). In this approach, the output model is a $3L \times 3T$ matrix W which makes an error between $X_1 \cdot W$ and Y_1 very small, and the genotype score is computed as $Y_{\text{pred}} = X_2 \cdot W$. The accuracy is determined by the measuring the difference between Y_2 and Y_{pred} with a well-known accuracy measurement such as micro-AUC.

However, as we consider tens of thousands of individuals and target SNPs, the computational cost to deal with X_1, Y_1 and W in HE is too large. As a result, it has become necessary to find an alternative approach to reduce the HE operation cost. We start from the underlying heuristic assumption that each target SNP is mostly influenced by tag SNPs which are close to the target SNP with respect to the genomic distance. That is, we only consider the closest d variant genotypes for each target SNP.

Under this setting, we can reduce the super-large training data $(X,Y) \in \{0,1\}^{M_0 \times 3L} \times \{0,1\}^{M_0 \times 3T}$ to several small data $(X_{1i}, Y_{1i}) \in \{0,1\}^{M_0 \times 3d} \times \{0,1\}^{M_0 \times 3}$ for $1 \le i \le T$. Note that Y_{1i} corresponds to the i-th target SNP, and X_{1i} corresponds to a set of d consecutive tag SNPs that is closest to the i-th target SNP. As an analogue of the training data X_1 and Y_1, the test data X_2 and Y_2 are also divided into T matrices $(X_{2i}, Y_{2i}) \in \{0,1\}^{M \times 3d} \times \{0,1\}^{M \times 3}$. As a result, we train total T neural network models $W_i \in \mathbb{R}^{3d \times 3}$ from each (X_{1i}, Y_{1i}) for $1 \le i \le T$ separately, and we will predict Y_2 by applying each model on X_{2i} for each i. Compared to the above naive approach, the size of small weight matrices $\{W_i\}_{1 \le i \le T}$ is L/d times smaller than $W \in \mathbb{R}^{3L \times 3T}$, so we can expect L/d times faster performance of the evaluation phase.

Selection of the closest d tag SNPs for each target SNP The term "the closest d tag SNPs" is actually ambiguous since the distance between a single (target) SNP and multiple (tag) SNPs has not been defined yet. For each target SNP, let d_i

be the genomic distance between the target SNP and the i-th tag SNP. We define the closest d tag SNPs to the target SNP as $i,(i+1),\dots,(i+d-1)$-th tag SNPs for a certain choice of i which minimizes $\max_{0\le j\le d-1} d_{i+j} = \max\{d_i, d_{i+d-1}\}$. An algorithm to find such i is very simple: Starting from $i=1$, compute $\max\{d_i, d_{i+d-1}\}$. If this max value is larger than the previous max value for $i-1$ then $i \leftarrow i+1$, and else stop. Note that this simple algorithm is based on the fact that $\max\{d_i, d_{i+d-1}\}$ has a unique local minima with respect to i. Repeat this procedure for all the target SNPs, then we can obtain the closest d tag SNPs for every target SNP.

Genotype Score Computation

Let $W_i \in \mathbb{R}^{3d\times 3}$ be a (pre-computed) weight matrix that were trained based on (X_{1i}, Y_{1i}) for $1\le i\le T$. Then our computation over encrypted data comes down to multiplying matrices $X_{2i}\cdot W_i$ for each i. Here, we use the following formula:

$$X_{2i}\cdot W_i = \sum_{r=1}^{3d} [X_{2i}]^r \odot [W_i]_r,$$

where $[\cdot]^r$ denotes the r-th column, $[\cdot]_r$ denotes the r-th row, and $\odot$ means the Hadamard multiplication.

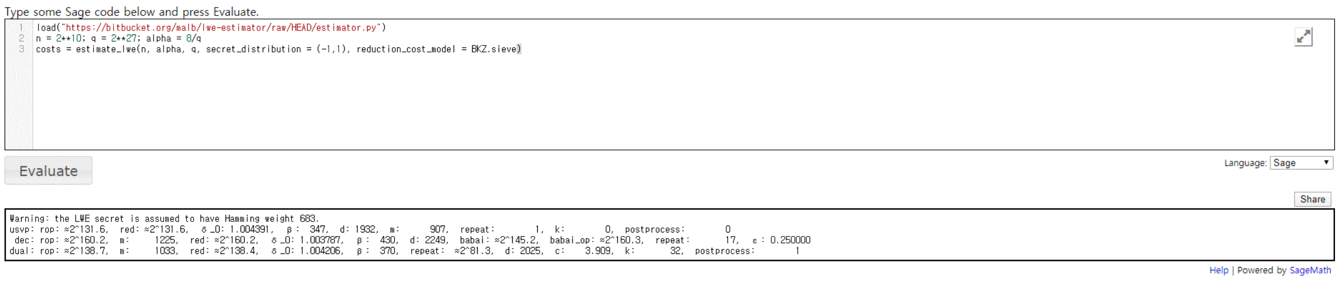
We encrypt each $[X_{2i}]^r$ separately for $1\le r\le 3d$ and $1\le i\le T$, then each multiplication $[X_{2i}]^r\cdot[W_i]_r$ is done by just a depth-1 constant multiplication. (Here, W_i's are encoded as CKKS plaintexts, so actually the multiplications are just constant multiplications.) As a result, our genotype score computation process consists of a number of depth-1 constant multiplications without any additional operations such as rotation.

Optimization Using Imaginary Parts Technically, we exploit imaginary parts of CKKS plaintext slots to enhance the speed of genotype score computation. When we compute Hadamard multiplication between two vectors $(\vec{x}_1,\vec{y}_1)$ and $(\vec{x}_2,\vec{y}_2)$, the summation of two matrix multiplications $\vec{x}_1\odot\vec{y}_1+\vec{x}_2\odot\vec{y}_2$ is expressed as the real part of the single multiplication $(\vec{x}_1+\sqrt{-1}\cdot\vec{x}_2)\odot(\vec{y}_1-\sqrt{-1}\cdot\vec{y}_2)$. This optimization enables us to reduce the number of matrix multiplications in $\sum_{r=1}^{3d}[X_{2i}]^r\odot[W_i]_r$ and the number of ciphertexts to encrypt X_{2i} by half since two columns can be encrypted in one ciphertext.

Experimental Results

CKKS parameters. As we only require depth 1, we can take extremely small CKKS parameters. We used CKKS parameters $n=2^{10}$, $q=2^{27}$, $\sigma=3.2$ with (uniformly random) signed binary secret, which satisfy more than 128-bit security according to Albrecht's LWE estimator[3]. We choose the scaling factor $\Delta=2^{17}$ for the encryption of X_2 and $\Delta=2^5$ for the plain models $\{W_i\}_{1\le i\le T}$. Note that we set the Hamming weight of the secret by $h=\lfloor 2n/3\rfloor$ in our code, and this corresponds to (uniformly random) signed binary case: In fact, the estimator outputs the same bit-security for both cases.

Security estimation on our parameter choice, according to lwe-estimator


Type some Sage code below and press Evaluate.

```
load("https://bitbucket.org/malb/lwe-estimator/raw/HEAD/estimator.py")
n = 2**10; q = 2**27; alpha = 8/q
costs = estimate_lwe(n, alpha, q, secret_distribution = (-1,1), reduction_cost_model = BKZ.sieve)
```

```
Warning: the LWE secret is assumed to have Hamming weight 683.
usvp: rop: ≈2^131.6,  red: ≈2^131.6,  δ_0: 1.004391,  β:  347,  d: 1932,  m:      907,  repeat:       1,  k:       0,  postprocess:       0
 dec: rop: ≈2^160.2,  m:     1225,  red: ≈2^160.2,  δ_0: 1.003787,  β:  430,  d: 2249,  babai: ≈2^145.2,  babai_op: ≈2^160.3,  repeat:      17,  ε: 0.250000
dual: rop: ≈2^138.7,  m:     1033,  red: ≈2^138.4,  δ_0: 1.004206,  β:  370,  repeat:  ≈2^81.3,  d: 2025,  c:   3.909,  k:      32,  postprocess:       1
```

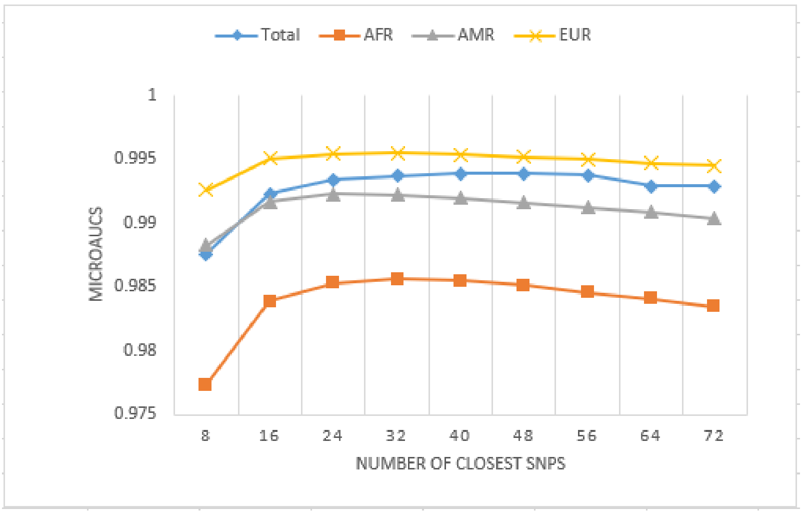
Experiments Based on HE. Table 5 shows the experimental results on our genotype imputation algorithm with various choices on the number of target SNPs (T =20K, 40K or 80K) and the number of closed tag SNPs ($d=8k$ for $1\le k\le 12$). Note that the accuracy is measured by micro-AUC. The larger d implies the higher computational cost, but it does not guarantee a higher accuracy. From Table 5, we can see that $d=40$ shows the best accuracy among $d=8k$ choices. When we set $d=8$, the accuracy somewhat decreases compared to the case $d=40$, but it shows about five times faster performance. Finally, the intermediate choice $d=24$ shows a balanced performance in terms of both accuracy and running time. As a result, we set $d=8$, 24 and 40 to be "fast", "balanced" and "best" choice of the number of closest tag SNPs, respectively.

All the experiments were performed on Linux operating systems on a workstation with Intel Platinum 8168 2.7GHz CPU with a 16-thread environment, compiled with GNU C++ 7.5.0 using the '-Ofast' optimization setting.

[3]Available at `https://bitbucket.org/malb/lwe-estimator`.

Supplementary Table 5. Experimental results of HE-based genotype imputation – SNU Team.

Scheme	Number of target SNPs	Number of closest tag SNPs	Time					Memory	Accuracy
			KeyGen	Encryption	Evaluation	Decryption	Total		
		8			1.354 sec		4.381 sec	5.121 GB	0.9831
		16			1.909 sec		4.936 sec	5.14 GB	0.9884
		24			2.594 sec		5.621 sec	5.158 GB	0.9895
		32			3.103 sec		6.130 sec	5.176 GB	0.9900
	20K	**40**	15.231 ms	1.908 sec	**3.835** sec	1.104 sec	**6.862** sec	5.195 GB	**0.9902**
		48			4.515 sec		7.542 sec	5.213 GB	0.9902
		56			5.384 sec		8.411 sec	5.231 GB	0.9900
		64			6.008 sec		9.035 sec	5.25 GB	0.9891
		72			7.227 sec		10.254 sec	5.268 GB	0.9890
		8			2.685 sec		6.593 sec	7.976 GB	0.9867
		16			3.525 sec		7.433 sec	8.012 GB	0.9918
		24			5.145 sec		9.053 sec	8.049 GB	0.9929
		32			6.093 sec		10.001 sec	8.086 GB	0.9934
CKKS	40K	**40**	15.231 ms	1.908 sec	**7.669 sec**	1.985 sec	**11.577 sec**	8.122 GB	**0.9949**
		48			9.167 sec		13.075 sec	8.159 GB	0.9935
		56			10.667 sec		14.575 sec	8.195 GB	0.9934
		64			11.947 sec		15.855 sec	8.232 GB	0.9926
		72			13.301 sec		17.209 sec	8.269 GB	0.9926
		8			4.762 sec		10.841 sec	13.81 GB	0.9876
		16			7.091 sec		13.170 sec	13.884 GB	0.9923
		24			9.582 sec		15.661 sec	13.958 GB	0.9934
		32			12.244 sec		18.323 sec	14.032 GB	0.9937
	80K	**40**	15.231 ms	1.908 sec	**15.167 sec**	4.156 sec	**21.246 sec**	14.106 GB	**0.9939**
		48			18.261 sec		24.340 sec	14.18 GB	0.9939
		56			21.009 sec		27.088 sec	14.254 GB	0.9938
		64			24.092 sec		30.171 sec	14.328 GB	0.9929
		72			26.934 sec		33.013 sec	14.402 GB	0.9929



Supplementary Figure 3. microAUC precision versus number of closest SNPs for different populations – SNU Team.

EPFL-Lattigo Team Solution

In this section, we explain the EPFL solution that relies on a multinomial logistic regression model to perform genotype imputation on homomorphically encrypted data. Our solution is built on the publicly available Lattigo Library[18] that notably offers centralized and distributed (multiparty) implementations of the BFV[6] and CKKS[19] schemes. Here, we use Lattigo's optimized implementation of the full-RNS variant of the centralized CKKS scheme.

Algorithm

Our goal is to design an efficient and accurate HE method for imputing missing entries in a given variant-genotype dataset. A SNP entry can take three different values: 0 for homozygous reference (0|0), 1 for heterozygous (1|0 or 0|1) and 2 for homozygous alternate (1|1). We train a multinomial logistic regression model, i.e., the model is trained jointly for the three label classes, for each position in which there are missing entries, by using a publicly available population panel.

The model is trained on plaintext data and is then stored in plaintext at a semi-honest server that provides prediction-as-a-service on homomorphically encrypted input data. In order to impute or predict a missing variant genotype, the model relies on the genotypes of variants in a d-sized hood of the missing entry (i.e., the closest $d-1$ variant genotypes).

Training on Plaintext Data. The logistic regression model is trained on a plaintext population panel for each of the T target positions, i.e., corresponding to missing entries for patients for which the imputation has to be performed. This panel is either publicly available, or available to the server that performs the training. We therefore obtain a vector $\mathbf{w}^{(\cdot,j,g)} = [w_0^{(\cdot,j,g)}, \dots, w_{d-1}^{(\cdot,j,g)}]$ of d regression coefficients for each target position $j = 1, 2, \dots, T$ and each label class, i.e., genotype $g \in \{0,1,2\}$.

Prediction on Encrypted Data. To perform the imputation on encrypted data, we use a full-RNS variant of the CKKS scheme, which enables arithmetic over $\mathbb{C}^{n/2}$. The CKKS parameters are denoted by the tuple (n, L, σ, Δ), where n, a power of two, is the degree of the cyclotomic polynomial that defines the used ring, L is the maximum level of a ciphertext, σ is the standard deviation of the noise distribution, and Δ is the plaintext scale that we choose also as a power of two. The plaintext and ciphertext spaces share the same domain and are represented by polynomials in $R_{Q_\ell} = \mathbb{Z}_{Q_\ell}[X]/(X^n+1)$, with $Q_\ell = \Pi_{i\in\{0,\dots,\ell\}} q_i$ for $0 \le \ell \le L$ where each q_i is a prime with $q_i \equiv 1 \pmod{2n}$. We can use two encodings: the map $\mathbb{C}^{n/2} \to R_Q$ (slot packing) enables a plaintext to encode up to $n/2$ complex values with component-wise additions and multiplications, whereas the map $\mathbb{R}^n \to R_Q$ (coefficient packing) enables encoding up to n real values in a plaintext, with a component-wise addition and multiplication by a scalar. We use the latter encoding, as component-wise products are not required in our solution and coefficient packing achieves a higher packing efficiency. Homomorphic operations are carried across all the values in a single instruction, multiple data (SIMD) fashion.

For this task, we consider a set of P patients, each having the same T missing genotype variants (i.e., $T = 20k, 40k$ or $80k$ in the main example). The server has a pre-trained model of d regression coefficients for each of the T target positions and for each class label (i.e., $g \in \{0,1,2\}$).

For each target position j and for each patient p, the probability for each label is obtained by computing:

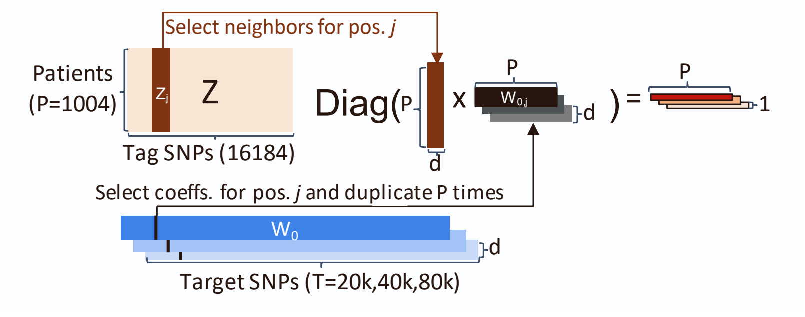
$$P[y = g|z_r^{(p,j)}] = \frac{e^{w_0^{(\cdot,j,g)} + \sum_{r=1}^{d-1} w_r^{(\cdot,j,g)}}}{\sum_{i=0}^{2} e^{w_0^{(\cdot,j,i)} + \sum_{r=1}^{d-1} w_r^{(\cdot,j,i)} z_r^{(p,j,\cdot)}}}, \tag{1}$$

where $\{w_0^{(\cdot,j,g)}, \dots, w_{d-1}^{(\cdot,j,g)}\}$ are the regression coefficients for label g and position j, and $\{z_1^{(p,j,\cdot)}, \dots, z_{d-1}^{(p,j,\cdot)}\}$ are the ing variants for patient p around target position j. These variants are encrypted, and the exponents $w_0^{(\cdot,j,g)} + \sum_{r=1}^{d-1} w_r^{(\cdot,j,g)} z_r^{(p,j,\cdot)}$ are computed in the encrypted domain, such that the result can be decrypted and used in the computation of the soft-max function. The (hard) prediction $y^{(p,j,g)}$ for $g \in \{0,1,2\}, p = 1,\dots,P$ and $j = 1,\dots,T$ can also be obtained by taking $y^{(p,j,g)} = \text{argmax}_g(P[y = g|z_r^{(p,j)}])$.

We now describe how to simultaneously perform the predictions for P patients for a specific position j by expressing the computations in matrix operations and relying on SIMD, as graphically depicted in Figure 4. The plaintext-trained models for each label compose three matrices W_0, W_1 and W_2 that are represented in blue in Figure 4. Each matrix has T columns and d rows. The input tag SNPs are represented in a matrix Z of P rows (patients) and 16184 (number of tag SNPs) columns. Z is encrypted column-wise, i.e., each column is a ciphertext.

The predictions for P patients for a specific target position j is performed by first selecting a sub-matrix Z_j of Z containing d columns corresponding to the d closest s of position j, and by selecting the corresponding column j in each W_g, $\forall\, g \in \{0,1,2\}$. The selected columns are duplicated P times to form matrices $W_{g,j}$ for each $g \in \{0,1,2\}$ of d rows and P columns. The rows of these matrices are each encoded in plaintext.

For each target position j, the multiplications between the rows of Z_j and the columns of $W_{g,j}$ are concurrently performed, such that the resulting three ciphertexts (i.e., one for each label) can be decrypted. The resulting plaintexts are then processed through the soft-max function, and the maximum value at each position, among the three labels, determines the predicted label for patient p at position j.


Select neighbors for pos. j
Patients (P=1004)
Z
Tag SNPs (16184)
Diag(
P
d
x
W0j
d
) =
1
Select coeffs. for pos. j and duplicate P times
W0
d
Target SNPs (T=20k,40k,80k)

Supplementary Figure 4. Predictions for T target positions in P patients.

Optimizations

We optimize the encryption of patient input data, its compression, and the encoding of the regression coefficients, to minimize storage needs, bandwidth and computation overhead. We use different scale factors Δ_{Patient} for the patient input data (which are already integers) and the coefficients Δ_{Model} .

Encryption of Input Tag SNPs Matrix Z. The matrix Z has P rows and 16184 columns, i.e., each column contains the SNPs of all P patients for a position j. We pack and encrypt each column in one ciphertext, i.e., Z is made of 16184 ciphertexts, each encrypting $P = 1004$ values. The prediction is performed by using plaintext model coefficients (clear-text values for the imputation server) that are equal for all patients, hence only ciphertext multiplications by scalar plaintexts are required (no polynomial products are needed). Thus, we do not need to use slot packing $\mathbb{C}^{n/2} \rightarrow R_Q[X]/(X^n+1)$ of the CKKS scheme and can directly encode the columns of Z in the coefficients of polynomials in R_Q; instead, we use a coefficient packing approach. Additions and multiplications by a scalar are commutative between the time and frequency (coefficients and slots) domains. The encoding therefore consists of a mapping from $\mathbb{R}[X]/(X^n+1) \rightarrow \mathbb{Z}_Q[X]/(X^n+1)$ that is simply done by scaling the coefficients by a plaintext scale Δ_{Patient}, rounding them and taking them modulo Q. The inverse mapping is simply the inverse scaling. This enables us to pack twice as many values per ciphertext, i.e., n instead of $n/2$. This reduces the data and algorithmic complexity during all the steps (encryption, prediction and decryption) by a factor of two, compared to using the default CKKS slot-encoding approach. It also improves the plaintext precision, as no approximate Fast Fourier Transform (FFT) is needed for encoding and decoding. Finally, as the client is the one encrypting the inputs and the owner of the secret key, we perform all encryptions with the secret key, therefore reducing the noise of fresh ciphertexts.

Client-Server Communication Each row of the Z matrix can be encrypted in a single ciphertext, providing that the ring degree n is large enough. In our setting, $P = 1004$ and $n = 1024$, hence the encryption of the matrix generates 16184 ciphertexts. Since a fresh encrypted ciphertext is of the form $(-\mathrm{a} \cdot \mathrm{sk} + m + e, a)$, the a polynomial can be generated in a deterministic way using a seeded pseudo-random number generator (PRNG) and only the seed needs to be sent to the server that can then reconstruct the polynomial. This further reduces the amount of data to be sent to the server by a factor of two. Here, each coefficient can be effectively represented by a uint32 that can be stored in four bytes; the PRNG seeds have a size of 64 bytes each. Furthermore, we can use a single seed per encryption instance (instead of one per ciphertext). The total size of the fresh encrypted data is thus $16184 \cdot 1024 \cdot 4 + k \cdot 64$ bytes where k is the number of encryption instances running in parallel. This amounts to a total of approximately 66 MB. The server prediction data comprises $3 \cdot$ TargetSNPs ciphertexts, which, for the largest setting (80k), amounts to $3 \cdot 80882 \cdot 2 \cdot 1024 \cdot 4$ bytes; i.e., approximately 2GB.

Prediction The model coefficients matrices $W_{g,j}$ for a target position j and for $g \in \{0,1,2\}$, each have d rows and P columns. Each row is encoded in a plaintext of identical values and is then multiplied with a ciphertext in the prediction. This multiplication can therefore be reduced to a multiplication by a plaintext scalar. Moreover, to facilitate constant additions (addition of $w_0^{(\cdot,j,g)}$ in Equation 1) and to avoid the need of computing a plaintext NTT for every prediction, the server pre-computes a plaintext in the NTT domain of the ciphertext with all coefficients set to one (and scaled by the factor Δ_{Patient}). It can then be used as a basis for all the subsequent constant additions, as it effectively switches the problem from a constant addition in the NTT domain back to a constant multiplication in the NTT domain. Finally, the coefficients $W_{g,j}$ of the model needed for a specific prediction are mapped from $\mathbb{R}$ to $\mathbb{Z}_Q$ by computing $\lceil \Delta_{\text{Model}} W_{g,j} \rfloor \mod Q$. We then avoid all modular reductions during the rest of the prediction by using a 29-bit prime for the modulus and performing all computations within uint64 variables. This enables us to do up to 64 (2^6) successive constant multiplications and additions without a risk of uint64 overflow because $(2^{29}-1) \cdot (2^{29}-1) \cdot 2^6 < 2^{64}$. As the maximal window size d that we use is 64, the modular reduction can be done once after the d-1 additions.

Supplementary Table 6. Experimental results of HE-based genotype imputation – EPFL Team.

Scheme	Number of target SNPs	Number of closest tag SNPs	Time (sec) KeyGen	Encryption	Evaluation	Decryption	Total	Memory (GB)
CKKS	20K	4	0.001	0.17	0.37	0.42	0.96	2.23
		8			0.49		1.08	
		16			0.54		1.13	
		32			0.72		1.31	
		48			1.14		1.73	
		64			1.49		2.08	
	40K	4	0.001	0.17	0.76	0.76	1.69	4.39
		8			1.01		1.94	
		16			1.26		2.19	
		32			1.43		2.36	
		48			2.24		3.17	
		64			2.95		3.88	
	80K	4	0.001	0.17	1.80	2.26	4.23	7.38
		8			1.95		4.38	
		16			2.58		5.01	
		32			3.24		5.67	
		48			4.67		7.10	
		64			6.10		8.53	

Supplementary Table 7. Accuracy metrics – EPFL Team.

Window size	4	8	16	32	48	64
Accuracy	0.8509	0.9140	0.9422	0.9516	0.9546	0.9547
Micro AUC	0.9659	0.9867	0.9929	0.9935	0.9950	0.9945

Parameters and Security

We describe here the parametrization for the prediction model and for the cryptosystem, to achieve the required level of bit-security. The plaintext scale for the patient data is $\Delta_{\text{Patient}} = 2^{16}$ and the plaintext scale for the model data is $\Delta_{\text{Model}} = 2^7$.

Cryptographic Parameters We use the ring degree $n = 2^{10}$, as it enables the required homomorphic capacity with a security level of at least 128 bits. As all ciphertexts are encrypted using the secret key and no homomorphic multiplication is needed, the distribution of the secret key has no impact on the underlying error. For this reason, both the secret-key and error polynomial are sampled from χ_{err}, which is a truncated Gaussian distribution with the standard deviation $\sigma = 3.2$ and a bound set to $\lfloor 6\sigma \rfloor$. This enables increasing the modulus from 27 bits (with a dense ternary key) to 29 bits (with Gaussian key), while still ensuring a security of 128 bits, according to the Homomorphic Encryption Standardization whitepaper[20]. Consequently, it enables a larger Δ_{Model} that also increases the precision of the prediction.

Experimental Results

Our source code is developed in GoLang, and builds on top of the Lattigo Library[18] and its optimized implementation of the full-RNS variant of the centralized CKKS scheme. Our experiments were performed on a machine with Intel Xeon Platinum 8168 CPU at 2.7 GHz with 769 GB RAM.Table 6 presents the performance of our solution for the genotype imputation of $T = 20k$, $40k$ and $80k$ SNPs for $P = 1004$ patients and different window sizes. As in the main paper, we limited the number of threads to 16. It must be noted that our solution is parallelized at multiple levels. It leverages the GoLang capabilities to execute the encryption, prediction and decryption in parallel across several ciphertexts, and it relies on the SIMD capability of the encryption scheme to parallelize the intra-ciphertext operations.

We obtain the same accuracy when the predictions are performed on encrypted data or on cleartext data, i.e., there is no precision loss. Table 7 reports the accuracy achieved with different window sizes, when the model is trained and tested on all the patients (ALL) or on a specific part of the population (African AFR, American AMR and European EUR patients).

UTHealth-Microsoft Research Team Solution

In this section, we explain the secure outsourced genotype imputation protocols developed by UT Health Science Center at Houston (UTHealth) and Microsoft Research (MSR). We provide implementations of linear imputation models by using two different HE frameworks – BFV[5,6] and CKKS[7]. Our secure imputation protocols are implemented in C++ with Microsoft SEAL version 3.4[21], which includes implementations of BFV and CKKS. Our code can be downloaded from https://github.com/K-miran/UTMSR_HEmpute.

Methods

The model in the current study is a linear model where each variant genotype is modeled using genotypes of tag variants within k variant vicinity of the target variant (i.e., genotypes of the closest $d = 2k$ tag variants to the target variant). We used the Gnu Scientific Library (GSL) library[4] to train the linear model weights.

Plaintext version. The tag variants of M samples is represented as a matrix $\mathbf{G} = (g_{i,\ell}) \in \{0,1,2\}^{M\times L}$. Let $\mathbf{g}_\ell$ be the ℓ^{th} column of the matrix $\mathbf{G}$. For $j = 1,2,\ldots,T$, we let $\{\mathbf{g}_{j_1},\ldots,\mathbf{g}_{j_d}\}$ be a set of the closet d variant genotypes of the j^{th} target variant and denote the learned model coefficients by $\mathbf{w}_j = (w_{j_0}, w_{j_1},\ldots,w_{j_d})$. Then the linear model can be used to estimate the missing genotypes by computing the column vector

$$\tilde{\mathbf{g}}_j = (\tilde{g}_{i,j})_{i=1}^{M} = \mathbf{w}_{j_0} + \sum_{r=1}^{d} w_{j_r}\cdot \mathbf{g}_{j_r} \in \mathbb{R}^M, \tag{2}$$

where $\tilde{g}_{i,j}$ represents the imputed genotype estimate of the i^{th} sample for the j^{th} target variant and $\mathbf{w}_{j_0}$ is a column vector obtained by making M copies of the intercept term w_{j_0}.

Ciphertext Version. The tag variant genotypes are column-wise encoded as polynomials with its coefficients $\mathsf{pt.g}_\ell = \sum_{i=0}^{M-1} g_{i,\ell}X^i$ for $\ell = 1,\ldots,L$ and then encrypted as a ciphertext $\mathsf{ct.g}_\ell$ using an underlying HE scheme. In the BFV scheme, the inner product between an encryption ct of a plaintext pt and the secret s of the HE scheme has the form of

$$\langle \mathsf{ct},\ \mathsf{s}\rangle = \lfloor q/p \rfloor \cdot \mathsf{pt} + \mathsf{e} \pmod{q}$$

for the plaintext modulus p and some small error polynomial e, so that a real message can be easily recovered by dividing $\lfloor q/p \rfloor$ and rounding the result. Here, we denote by $[\cdot]_q$ reduction modulo q into the interval $[0,q)$. Whilst, in the CKKS scheme, it has the form of

$$\langle \mathsf{ct},\ \mathsf{s}\rangle = \mathsf{pt} + \mathsf{e} \pmod{q}$$

for some small error e, which is an approximate value of the plaintext. This encoding procedure results in some loss of precision, hence plaintext values should be multiplied by a scaling factor of Δ_{tag} to ensure that the encoded values retain enough precision.

Our underlying HE schemes deal only with integers, hence we should convert the real-valued model parameters into finite-precision integers. Let Δ_{wt} be a pre-determined quantization parameter for the weight parameters. In the BFV scheme, each slope coefficient of the parameters is converted into an element in the ring R/pR by taking $\mathsf{pt.w}_{j_r} = [\Delta_{\mathsf{wt}}\cdot w_{j_r}]_p$ for $r = 1,\ldots,d$. In particular, the intercept term is encoded as an integral polynomial $\mathsf{pt.w}_{j_0} = \sum_{i=0}^{M-1}[\Delta_{\mathsf{wt}}\cdot w_{j_0}]_p X^i$.

In the context of the CKKS scheme, the slope coefficient is transformed into an element in the ring R/qR by computing $\mathsf{pt.w}_{j_r} = [\Delta_{\mathsf{wt}}\cdot w_{j_r}]_q$. As mentioned above, the tag variants are multiplied by a factor of Δ_{tag} before encryption, so the intercept is encoded as $\mathsf{pt.w}_{j_0} = \sum_{i=0}^{M-1}[\Delta_{\mathsf{wt}_0}\cdot w_{j_0}]_q X^i$ for $\Delta_{\mathsf{wt}_0} := \Delta_{\mathsf{wt}}\cdot\Delta_{\mathsf{tag}}$ in order to keep their common scaling factor of the results.

After the data owner encrypts the genotype plaintext and sends them to the server, the server performs constant addition and multiplication operations on the encrypted genotype data by using the encoded weight polynomials. To be precise, the j^{th} imputed genotype estimates $\tilde{\mathbf{g}}_j$ can be securely computed by

$$\mathsf{pt.w}_{j_0} + \sum_{r=1}^{d} \mathsf{pt.w}_{j_r}\cdot \mathsf{ct.g}_{j_r},$$

where $\mathsf{ct.g}_{j_r}$ represents an encryption of the plaintext $\mathsf{pt.g}_{j_r}$. Algorithm 2 and 3 provide explicit descriptions of homomorphic evaluation of genotype imputation based on the BFV and CKKS schemes, respectively.

[4] http://www.gnu.org/software/gsl/

Algorithm 2 Secure genotype imputation protocol based on the BFV scheme.

```
Require: Ciphertexts of tag variants $\{\mathsf{ct.g}_\ell\}_{1\le\ell\le L}$; regression coefficients $\{\mathbf{w}_j = (w_{j_0}, w_{j_1}, \dots, w_{j_d})\}_{1\le j\le T}$
Ensure: Encrypted genotype estimate values $\{\mathsf{ct.\tilde{g}}_j\}_{1\le j\le T}$
1: for $1 \le j \le T$ do
2:     for $1 \le r \le d$ do
3:         $\mathsf{pt.w}_{j_r} \leftarrow [\Delta_{\mathsf{wt}} \cdot w_{j_r}]_p$
4:     end for
5:     $\mathsf{pt.w}_{j_0} \leftarrow \sum_{i=0}^{M-1} [\Delta_{\mathsf{wt}} \cdot w_{j_0}]_p X^i$
6:     $\mathsf{ct.\tilde{g}}_j \leftarrow \mathsf{pt.w}_{j_0} + \sum_{r=1}^{d} (\mathsf{pt.w}_{j_r} \cdot \mathsf{ct.g}_{j_r})$
7: end for
```

Algorithm 3 Secure genotype imputation protocol based on the CKKS scheme.

```
Require: Ciphertexts of tag variants $\{\mathsf{ct.g}_\ell\}_{1\le\ell\le L}$; regression coefficients $\{\mathbf{w}_j = (w_{j_0}, w_{j_1}, \dots, w_{j_d})\}_{1\le j\le T}$; the scaling factor
    of tag variants $\Delta_{\mathsf{tag}}$
Ensure: Encrypted genotype estimate values $\{\mathsf{ct.\tilde{g}}_j\}_{1\le j\le T}$
1: for $1 \le j \le T$ do
2:     for $1 \le r \le d$ do
3:         $\mathsf{pt.w}_{j_r} \leftarrow [\Delta_{\mathsf{wt}} \cdot w_{j_r}]_q$
4:     end for
5:     $\mathsf{pt.w}_{j_0} \leftarrow \sum_{i=0}^{M-1} [\Delta_{\mathsf{wt}} \cdot \Delta_{\mathsf{tag}} \cdot w_{j_0}]_q X^i$
6:     $\mathsf{ct.\tilde{g}}_j \leftarrow \mathsf{pt.w}_{j_0} + \sum_{r=1}^{d} (\mathsf{pt.w}_{j_r} \cdot \mathsf{ct.g}_{j_r})$
7: end for
```

Correctness. In the BFV scheme, it follows from our construction of the encryption method that

$$\begin{aligned}
\Delta_{\mathsf{wt}} \cdot \left(\sum_{i=0}^{M-1} \tilde{g}_{i,j} X^i\right) &= \Delta_{\mathsf{wt}} \cdot \sum_{i=0}^{M-1} \left(w_{j_0} + \sum_{r=1}^{d} (w_{j_r} \cdot g_{i,j_r})\right) X^i \\
&= \sum_{i=0}^{M-1} (\Delta_{\mathsf{wt}} \cdot w_{j_0}) X^i + \sum_{r=1}^{d} \left((\Delta_{\mathsf{wt}} \cdot w_{j_r}) \cdot \left(\sum_{i=0}^{M-1} g_{i,j_r} X^i\right)\right) \\
&= \sum_{i=0}^{M-1} (\Delta_{\mathsf{wt}} \cdot w_{j_0}) X^i + \sum_{r=1}^{d} (\Delta_{\mathsf{wt}} \cdot w_{j_r}) \cdot \mathsf{pt.g}_{j_r},
\end{aligned}$$

where the first equality follows from Eq. (2). This implies that the i^{th} coefficient of the plaintext polynomial obtained from decryption is the scaled predicted value $\tilde{g}_{ij}$ under an appropriate valid parameter selection. Similarly, in the CKKS scheme, we have

$$\begin{aligned}
\Delta_{\mathsf{wt}_0} \cdot \left(\sum_{i=0}^{M-1} \tilde{g}_{i,j} X^i\right) &= \Delta_{\mathsf{wt}_0} \cdot \sum_{i=0}^{M-1} \left(w_{j_0} + \sum_{r=1}^{d} (w_{j_r} \cdot g_{i,j_r})\right) X^i \\
&= \sum_{i=0}^{M-1} (\Delta_{\mathsf{wt}_0} \cdot w_{j0}) X^i + \sum_{r=1}^{d} \left((\Delta_{\mathsf{wt}} \cdot w_{jr}) \cdot \left(\sum_{i=0}^{M-1} (\Delta_{\mathsf{tag}} \cdot g_{i,j_r}) X^i\right)\right) \\
&= \sum_{i=0}^{M-1} (\Delta_{\mathsf{wt}_0} \cdot w_{j_0}) X^i + \sum_{r=1}^{d} (\Delta_{\mathsf{wt}} \cdot w_{j_r}) \cdot \mathsf{pt.g}_{j_r}.
\end{aligned}$$

Therefore, the i^{th} coefficient of the plaintext polynomial obtained from decryption is approximate to the predicted value $\tilde{g}_{ij}$ scaled by the factor of Δ_{wt_0}.

Experiments

Parameters and Security. In our implementation, we used $\Delta_{\mathsf{wt}} = 6$ to encode the slope coefficients of the model parameters. Also, we chose $\Delta_{\mathsf{tag}} = 2^{16}$ to encode the tag variant genotypes in the CKKS scheme, so that it can reduce the precision loss in homomorphic computation. We note that all the operations on plaintext polynomials are performed modulo p in the BFV scheme, thus the final result should be less than the plaintext coefficient modulus p so that no reduction modulo p

Supplementary Table 8. Experimental results of secure genotype imputation protocols on the testing genotype dataset with 1,004 individuals in a 16-thread environment – UTHealth-MSR Team.

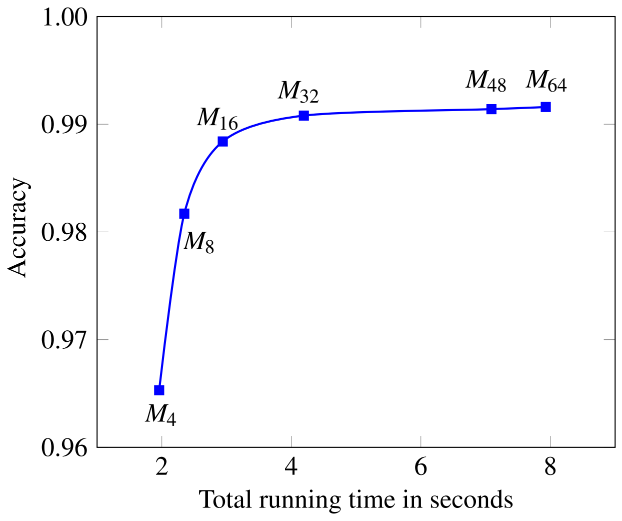
Scheme	# target variants	# tag variants as features	Time: Encryption	Time: Evaluation	Time: Decryption	Time: Total	Memory	Macro Accuracy: All-var	Macro Accuracy: Non-ref
BFV	20K	4	0.162 sec	0.204 sec	0.248 sec	0.614 sec	0.994 GB	0.8512	0.6712
		8		0.276 sec		0.686 sec	0.998 GB	0.8948	0.7771
		16		0.508 sec		0.918 sec	1.034 GB	0.9181	0.8360
		32		0.699 sec		1.109 sec	1.114 GB	0.9286	0.8629
		48		0.899 sec		1.308 sec	1.211 GB	0.9314	0.8701
		64		1.495 sec		1.905 sec	1.316 GB	0.9330	0.8746
	40K	4	0.326 sec	0.405 sec	0.541 sec	1.271 sec	1.826 GB	0.8634	0.6857
		8		0.594 sec		1.460 sec	1.832 GB	0.9099	0.8004
		16		0.919 sec		1.786 sec	1.873 GB	0.9349	0.8656
		32		1.477 sec		2.343 sec	1.947 GB	0.9463	0.8940
		48		1.766 sec		2.632 sec	2.026 GB	0.9493	0.9018
		64		3.397 sec		4.263 sec	2.098 GB	0.9511	0.9063
	80K	4	0.603 sec	0.851 sec	1.145 sec	2.599 sec	3.518 GB	0.8700	0.7042
		8		1.280 sec		**3.029 sec**	**3.539 GB**	**0.9153**	**0.8143**
		16		2.163 sec		3.912 sec	3.572 GB	0.9393	0.8753
		32		2.790 sec		**4.539 sec**	**3.654 GB**	**0.9502**	**0.9017**
		48		5.735 sec		7.484 sec	3.721 GB	0.9533	0.9090
		64		6.594 sec		**8.343 sec**	**3.802 GB**	**0.9550**	**0.9134**
CKKS	20K	4	0.150 sec	0.153 sec	0.161 sec	0.465 sec	1.003 GB	0.8506	0.6666
		8		0.228 sec		1.019 sec	1.007 GB	0.8944	0.7733
		16		0.436 sec		1.259 sec	1.020 GB	0.9177	0.8329
		32		0.703 sec		1.015 sec	1.109 GB	0.9283	0.8602
		48		0.907 sec		1.218 sec	1.194 GB	0.9311	0.8675
		64		1.561 sec		1.873 sec	1.281 GB	0.9328	0.8721
	40K	4	0.301 sec	0.305 sec	0.400 sec	1.006 sec	1.825 GB	0.8629	0.6810
		8		0.664 sec		1.179 sec	1.834 GB	0.9095	0.7968
		16		1.278 sec		1.619 sec	1.866 GB	0.9347	0.8628
		32		2.289 sec		1.887 sec	1.944 GB	0.9460	0.8918
		48		3.526 sec		2.145 sec	1.983 GB	0.9491	0.8997
		64		4.818 sec		3.989 sec	2.061 GB	0.9509	0.9043
	80K	4	0.603 sec	0.638 sec	0.812 sec	2.053 sec	3.517 GB	0.8696	0.6999
		8		1.013 sec		**2.427 sec**	**3.538 GB**	**0.9150**	**0.8109**
		16		1.616 sec		3.030 sec	3.572 GB	0.9390	0.8726
		32		2.798 sec		**4.212 sec**	**3.653 GB**	**0.9500**	**0.8995**
		48		5.546 sec		6.960 sec	3.721 GB	0.9531	0.9070
		64		6.362 sec		**7.776 sec**	**3.802 GB**	**0.9548**	**0.9114**

occurs. As a result, we took the plaintext modulus $p = 2^{10}$ to get the correct results after decryption. Next, in both cases, we used the ciphertext modulus $\log q = 27$ to guarantee the correctness of decryption in BFV or ensure that no overflow occurs during computation in CKKS. We choose the secret key from the uniform distribution over the set of polynomials in R whose coefficients are in $\{-1,0,1\}$. Each coefficient of an error is sampled from the discrete Gaussian distribution centered at zero with standard deviation of $\sigma = 3.2$. Finally, we took the ciphertext dimension $n = 2^{10}$ to provide at least 128-bit security level from the HE standardization workshop paper[20].

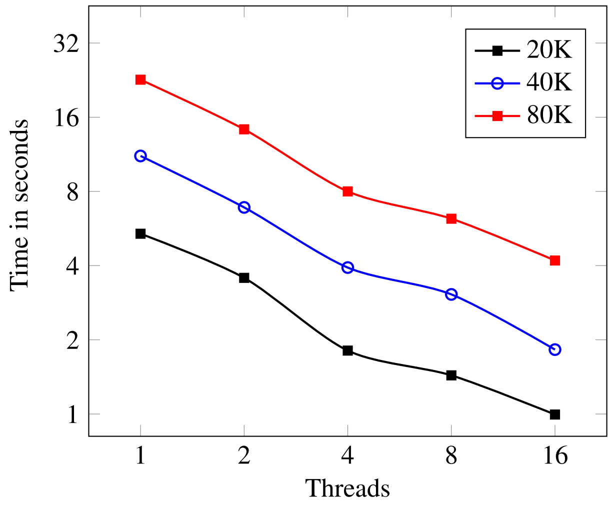
Experimental Results. Our experiments were conducted on a machine with Intel Platinum 8168 2.7GHz CPU in a 16-thread environment, compiled with GNU C++ 7.5.0 using the '-O3' optimization setting. Our secure imputation protocol takes around one millisecond for key generation. Table 8 presents the performance results for secure genotype imputation with various numbers of target variants (T) and various numbers of tag variants as features for each target variant (d). In the seventh and eighth columns, we present the total running time and the peak of the memory required for the secure imputation process, respectively. The last two columns give the macro-aggregated accuracies over all the predictions and non-reference genotype predictions. As seen in the table, the executive time of secure imputation process is almost linear on the target variant dataset size T. Overall, with the increasing number of predictors d, it results in a decrease in time performance but provides an increase in imputation accuracy.

Depending on the choice of the model, we can include a small or wide range of neighboring SNPs into consideration, and involve light or heavy arithmetic operations in the computation. As shown in Figure 5, such a choice will have two major effects: (1) the efficiency of the model and (2) the accuracy of the prediction. We estimated the prediction accuracy using micro-AUCs of the actual genotypes and the imputed genotypes. The accuracy of the prediction is around 0.9817 from the small-sized model (linear model with $d = 8$), 0.9908 from the medium-sized model (linear model with $d = 32$), and 0.9916 from the large-sized model (linear model with $d = 64$). Our experimental results show that the medium-sized model is optimal for timing and imputation accuracy in the current testing dataset. As a result, the best end-to-end performance of the most balanced model is approximately 52 microseconds per variant per 1000 individuals ($(4.212 \text{ sec} \times 1000 \text{ individuals})/(80,882 \text{ variants} \times 1004 \text{ individuals})$).

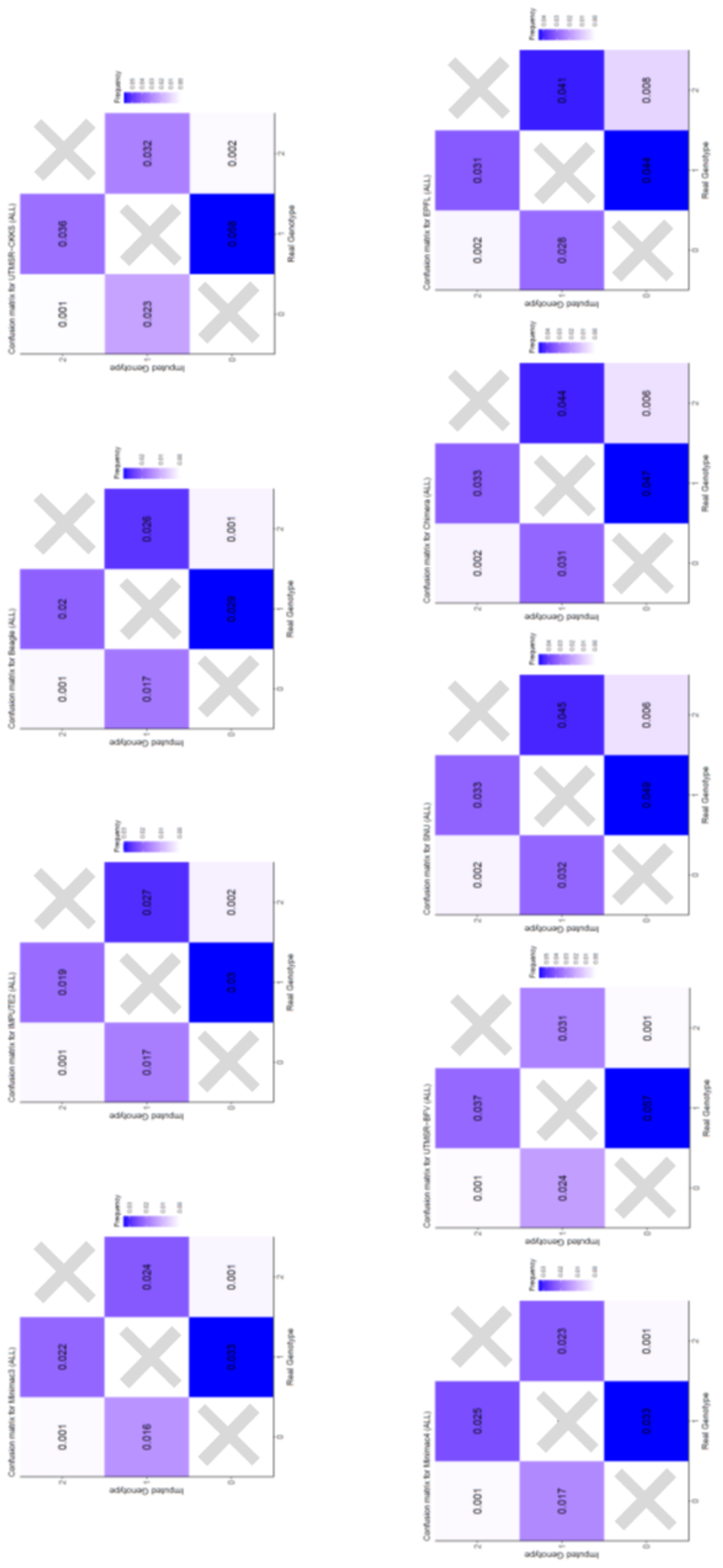
Figure 6 shows the timing results of the CKKS-based evaluation with various number of available threads. To show the scalability, we selected the most balanced model with 32 tag variants as features for each target variant. Our implementation exploits multiple cores, when available. And these results show that with at least up to 16 cores the speed-ups scale linearly with the number of cores.



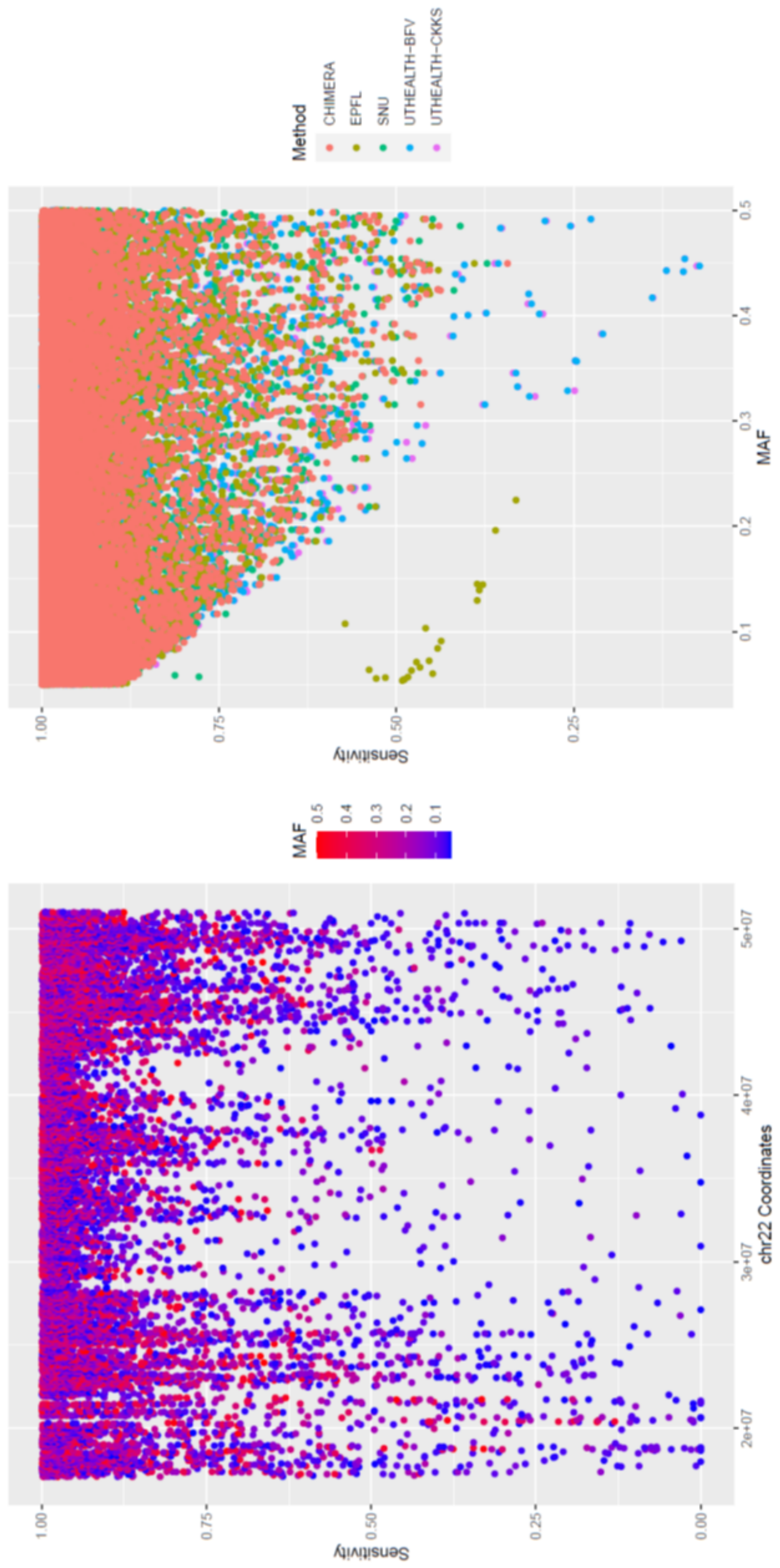
Supplementary Figure 5. The UTHealth-MSR team's CKKS implementation of the linear model M_d on the whole dataset (of 80K target variants) with respect to various numbers of tag variants at features for each target variants (d). The accuracy in y-axis means the micro-AUC of the predicted genotypes.



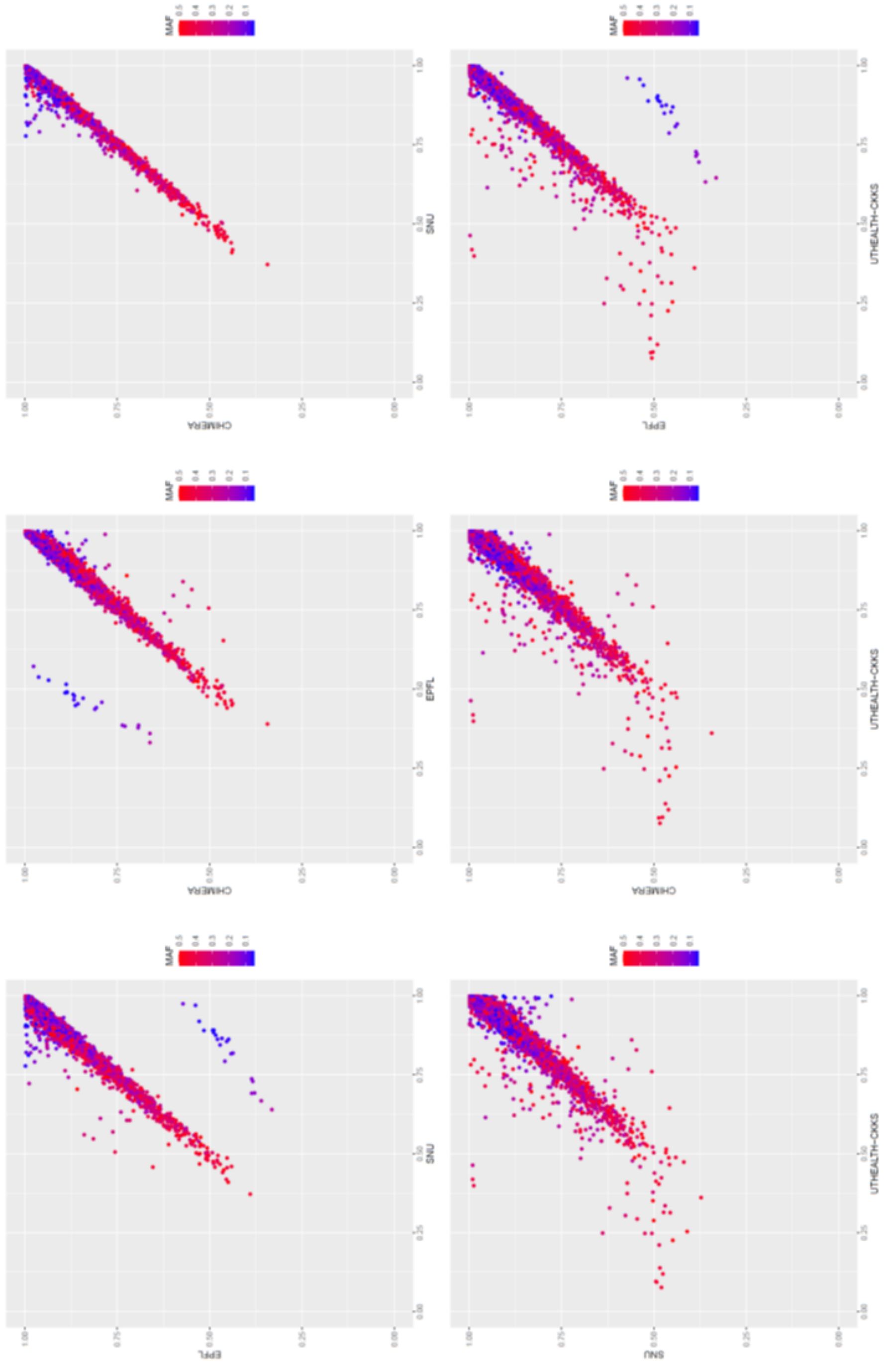
Supplementary Figure 6. Total running time result of the UTHealth-MSR team's CKKS implementation with respect to various target variant dataset size and threads (log-log scale) when using the medium-sized linear model (with 32 tag variants as features).



Supplementary Figure 7. The confusion matrices for the 8 benchmarked imputation methods. Each plot shows the confusion matrix for a method where x-axis shows the real genotype and y-axis shows the imputed genotype. Each cell contains the normalized frequency of confusions with respect to the total number of imputed genotypes, i.e. rows. The correctly predicted genotypes, i.e., the diagonals are censored intentionally (Marked with 'X') to make the colors more interpretable and comparable. The numeric value of each confusion entry is included for comparison of the confusion values.



Supplementary Figure 8a and 8b. Figure 8a shows the position (x-axis) vs genotype concordance (y-axis). Each point is a variant and color indicates the minor allele frequency. Figure 8b shows the MAF (x-axis) and genotype concordance for different secure methods, as indicated by each color.



Supplementary Figure 8c. Pairwise comparison of the secure methods. Each point represents an untyped (target) variant and colors indicate the minor allele frequency of the variants. The compared methods are indicated on the X-axis and Y-axis labels.

Supplementary Table 9. Rare Variant Imputation Performance Statistics for SNU-CKKS.

WindowSize	AUC	Seconds					Mbytes
		Keygen	Encryp.	Eval.	Decryp.	Total	Memory
8	0.9976	1.718	1933.38	9410.98	7741	19087.08	18227
16	0.9982	1.693	2002.71	16884.80	9212	28101.20	18227
24	0.9957	1.774	2116.67	23702.10	8233	34053.54	18227
32	0.9976	1.990	2127.51	29063.50	8145	39338.00	18227
40	0.9919	1.776	1999.89	35818.50	7930	45750.17	18227
48	0.9849	1.967	2135.28	45878.40	9088	57103.65	18227
56	0.9751	1.814	1997.14	51865.40	8284	61968.35	18227

Supplementary Table 10. Rare Variant Imputation Performance Statistics for Chimera-TFHE.

Window Size	Concordance	Seconds					Mbytes
		Keygen	Encryp.	Eval.	Decryp.	Total	Memory
5	0.9728	9.4E-5	1.7	3.5	1.6	6.8	3700
10	0.9772	9.4E-5	1.7	5.5	1.6	8.8	4100
15	0.9801	9.4E-5	1.7	7.5	1.6	10.8	4700
20	0.9819	9.4E-5	1.7	9.5	1.6	12.7	5000
25	0.9830	9.4E-5	1.7	11.3	1.6	14.6	5300
30	0.9837	9.4E-5	1.7	13.4	1.6	16.7	6100
35	0.9841	9.4E-5	1.7	15.6	1.6	18.9	6400
40	0.9844	9.4E-5	1.7	17.1	1.6	20.3	6700
45	0.9845	9.4E-5	1.7	19.0	1.6	22.2	7100
50	0.9847	9.4E-5	1.7	21.6	1.6	24.8	7400

References

1. Das, S. *et al.* Next-generation genotype imputation service and methods. *Nat. genetics* **48**, 1284–1287 (2016).

2. Loh, P. R. *et al.* Reference-based phasing using the haplotype reference consortium panel. *Nat. Genet.* **48**, 1443–1448, DOI: 10.1038/ng.3679 (2016).

3. Yu, Z. & Schaid, D. J. Methods to impute missing genotypes for population data. *Hum. Genet.* **122**, 495–504, DOI: 10.1007/s00439-007-0427-y (2007).

4. Harlemon, M. *et al.* A custom genotyping array reveals population-level heterogeneity for the genetic risks of prostate cancer and other cancers in africa. *Cancer Res.* **80**, 2956–2966, DOI: 10.1158/0008-5472.CAN-19-2165 (2020). https://cancerres.aacrjournals.org/content/80/13/2956.full.pdf.

5. Brakerski, Z. Fully homomorphic encryption without modulus switching from classical GapSVP. In Safavi-Naini, R. & Canetti, R. (eds.) *CRYPTO 2012*, vol. 7417 of *Lecture Notes in Computer Science*, 868–886 (Springer, 2012).

6. Fan, J. & Vercauteren, F. Somewhat practical fully homomorphic encryption. *IACR Cryptol. ePrint Arch.* **2012**, 144 (2012).

7. Cheon, J. H., Kim, A., Kim, M. & Song, Y. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, 409–437 (Springer, 2017).

8. Chillotti, I., Gama, N., Georgieva, M. & Izabachène, M. TFHE: Fast fully homomorphic encryption over the torus. *J. Cryptol.* (2019).

9. Abdalla, M., Benhamouda, F. & Pointcheval, D. Public-key encryption indistinguishable under plaintext-checkable attacks. In Katz, J. (ed.) *Public-Key Cryptography – PKC 2015*, 332–352 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2015).

10. Johnson, E. O. *et al.* Imputation across genotyping arrays for genome-wide association studies: Assessment of bias and a correction strategy. *Hum. Genet.* **132**, 509–522, DOI: 10.1007/s00439-013-1266-7 (2013).

11. Baza, M., Salazar, A., Mahmoud, M., Abdallah, M. & Akkaya, K. On sharing models instead of data using mimic learning for smart health applications (2019). 1912.11210.

12. Wojcik, G. *et al.* Imputation-aware tag snp selection to improve power for large-scale, multi-ethnic association studies. *G3* **8**, 3255–3267, DOI: 10.1534/g3.118.200502 (2018).

13. Browning, B. L. & Browning, S. R. Genotype imputation with millions of reference samples. *The Am. J. Hum. Genet.* **98**, 116–126, DOI: https://doi.org/10.1016/j.ajhg.2015.11.020 (2016).

14. Vowpal wabbit (fast learning). https://vowpalwabbit.org/research.html. Accessed: 2021-04-19.

15. Gentry, C. Fully homomorphic encryption using ideal lattices. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, 169–178, DOI: 10.1145/1536414.1536440 (ACM, 2009).

16. Boura, C., Gama, N. & Georgieva, M. Chimera: a unified framework for B/FV, TFHE and HEAAN fully homomorphic encryption and predictions for deep learning. *IACR Cryptol. ePrint Arch.* **2018**, 758 (2018).

17. Albrecht, M. R., Player, R. & Scott, S. On the concrete hardness of learning with errors. *J. Math. Cryptol.* **9**, 169–203 (2015).

18. Lattigo: A library for lattice-based homomorphic encryption in go. https://github.com/ldsec/lattigo (14.02.2019).

19. Cheon, J. H., Han, K., Kim, A., Kim, M. & Song, Y. A full RNS variant of approximate homomorphic encryption. In *Selected Areas in Cryptography (SAC 2018)* (2018).

20. Albrecht, M. *et al.* Homomorphic encryption security standard. Tech. Rep., HomomorphicEncryption.org, Toronto, Canada (2018).

21. Microsoft SEAL (release 3.4). https://github.com/Microsoft/SEAL (2019). Microsoft Research, Redmond, WA.