Homomorphic Encryption: An Application to Polygenic Risk Scores

Elizabeth Knight
elizabeth.knight@yale.edu

Jiaqi Li
jiaqi.li@yale.edu

Matthew Jensen
matthew.jensen@yale.edu

Israel Yolou
israel.yolou@yale.edu

Can Kockan
can.kockan@yale.edu

Mark Gerstein
pi@gersteinlab.org

May 26, 2024

Contents

1 Abstract **2**

2 Introduction **2**

3 Results **4**

3.1 Method Overview 4

3.1.1 Evaluation Strategy 4

3.2 Technical Overview of Key Aspects of FHE for PRS 5

3.3 Implementation Results 6

3.3.1 Overview of PRS derivations 6

3.4 Effects of Input Data Size on Runtime and Memory for Encrypted PRS 7

3.5 Encryption Parameters affect Accuracy and Runtime for PRS Calculation 8

4 Discussion **8**

5 Conclusion **9**

6 Methods **9**

6.1 CKKS Encryption Scheme 9

6.2 Key Generation 9

6.3 Implementation of Homomorphic Encryption 10

6.4 Implementation of Inner Product using Homomorphic Encryption 10

6.5 Synthetic Data Generation and Ridge Regression 11

6.6 Schizophrenia PRS generation 12

1 Abstract

Background: Polygenic risk scores (PRS) have emerged as a powerful tool in precision medicine, enabling personalized risk assessments for complex diseases. However, using sensitive genomic data in PRS calculations raises concerns about privacy and security. Fully Homomorphic Encryption (FHE) offers a promising solution by allowing computations on encrypted data, preserving the privacy of both genomic information and PRS models.

Results: In this study, we present a novel application of FHE for secure and private PRS calculations using the CKKS protocol within the Lattigo library. Our approach involves a three-party system: clients (doctors with sensitive genetic data), modelers developing a PRS (academics or a company), and evaluators (a "local hospital" running the models while maintaining data confidentiality). We demonstrate the feasibility and accuracy of our protocol by applying it to synthetic datasets of various sizes and a robust 110k-SNP model for schizophrenia. The normal PRS calculation results are essentially identical to the encrypted calculation: between the two results R^2 is .999 & MSE is 2.27×10^{-6}. Moreover, while the encrypted calculation is roughly 1000 times slower than conventional non-encrypted ones (when only considering the core PRS calculation), it is quite feasible on a single-CPU node - e.g. running on ˜1100 individuals with ˜110k SNPS took six minutes and ˜65G memory on a laptop computer. In addition, we investigate the impact of encryption parameters (modulus) on this computational time and accuracy in detail.

Conclusion: By enabling secure PRS calculations on encrypted genomic data, our approach addresses the pressing need for privacy-preserving solutions in the era of precision medicine. The ability to perform accurate risk assessments while maintaining patient confidentiality paves the way for broader adoption of PRS and personalized medicine in healthcare, particularly with the advent of large-scale computing power.

2 Introduction

Advances in genetic sequencing technology have significantly reduced time and cost, making it a commonly used practice in biomedical research and precision medicine [1]. Genomic sequencing data has the potential to provide valuable insights into biological processes and clinical treatment, especially with the multitude of associations established between genomic loci and traits or diseases through genome-wide association studies (GWAS) [2]. Such studies hold promise for various clinical applications, from more accurate molecular diagnosis to drug target prioritization [3] [4]. A meaningful clinical advance stemming from GWAS is the development of polygenic risk scores (PRS), which are statistical models that utilize genetic variants to predict an individual's risk of having a particular trait or condition, such as disease, height, or intelligence [5]. PRS has the potential to bring personalized approaches to health and wellness, from early detection and prevention through lifestyle changes or medical interventions to more effective and personalized care. PRS can also identify subgroups of individuals more likely to respond to certain medications or treatments, improving patient outcomes [6]. For example, a recent study found that PRS can indicate risk of future heart attacks, prioritizing patients for preventive statin treatments [7]. Similarly, PRS for breast cancer can identify subgroups of individuals with high predicted incidence for early screening interventions [8], further highlighting the utility of PRS in a clinical setting.

GWAS for most common traits (such as height) and diseases (such as breast cancer) typically require genomic data from tens of thousands of individuals at hundreds of thousands of loci for accurate results [9] [10]. This genetic data's sheer amount and complexity can strain local computational resources and often require the involvement of third parties or cloud-based infrastructures,

thereby increasing the risk of genetic privacy violations [11]. For example, the UKBiobank has over 400k samples which are often too massive to download fully, requiring the use of the DNANexus platform for downstream analysis [12]. In the face of quantum computing advancements, quantum-secure encryption has become crucial in ensuring the privacy of data transfers across systems [13]. As genomic data becomes increasingly important in a healthcare setting, hospitals need to adopt advanced encryption solutions to ensure the privacy and security of patient data, especially genomic data, while also reducing costs and improving the efficiency of clinical research. Protection of patient genomic data is particularly important for PRS calculation, where one individual's genome is compared against summary results derived from a large, often external cohort. Thus, advanced encryption methods are needed to protect sensitive genomic data from unauthorized access and misuse while allowing physicians to calculate and access their patients' PRS securely and confidentially. With the increasing use of PRS models trained on confidential patient information, establishing model security is crucial to protect clinical data and prevent potential model inversion attacks [14].

Homomorphic encryption is a promising solution to the problem of preserving confidentiality while allowing important computations on sensitive genomic data. This type of encryption allows third parties to perform computational functions on encrypted data (referred to as ciphertext) without first decrypting it. This addresses privacy concerns in digital communication by preserving confidentiality when manipulating data without exposing it to an unintended party [15]. There are several types of homomorphic encryption, including partially homomorphic, somewhat homomorphic, leveled fully homomorphic, and fully homomorphic (FHE), each offering varying levels of computation on encrypted data [15]. In brief, homomorphic encryption deliberately introduces a trivial amount of noise into the ciphertexts to ensure security; this noise increases with each computational operation performed (addition adds less noise than multiplication). However, different types of homomorphic encryption have limitations regarding the set of circuits, or the range and complexity of computational sequences that can be performed on encrypted data, and types of gates, or basic operations (like addition or multiplication) that are supported in these computations [16]. FHE introduces bootstrapping, a computationally expensive operation that reduces the noise of ciphertexts [17]. Bootstrapping thus allows for more computation with greater efficiency, and specific problems requiring minimal multiplication steps, such as applying PRS, perform well with this homomorphic implementation. As such, previous applications of FHE in healthcare have set the stage for advancing patient data privacy. For example, [18] initially leveraged this technology to handle vital sign data from medical devices securely. [19] later extended this use of FHE to protect genetic data within hospital systems, enabling researchers to access GWAS summary statistics without compromising patient privacy. A natural extension of this work to implement FHE in PRS calculation could significantly enhance data security by maintaining patient genomes in encrypted formats, potentially reducing vulnerabilities and costs when facilitating the secure transfer and manipulation of encrypted genomic data.

Here, we apply FHE specifically to genotype data for secure PRS calculation and phenotype prediction. Utilizing the CKKS protocol for FHE within the Lattigo library [20], we present a novel approach that allows for the computation of PRS directly on encrypted genetic datasets, obtaining results securely and privately. Our approach entails the propagation of encrypted data across three parties (clients with sensitive genetic data, modelers with existing PRS models, and evaluators to interpret PRS findings), preserving model and genomic privacy when communicating PRS results back to the patient [21]. This protocol was inspired by the iDASH 2022 secure genome analysis competition [21], and our solution addresses task 2, "Secure Model Evaluation on Homomorphically Encrypted Genotype Data." While other studies have proposed homomorphic encryption of PRS models and provided proof-of-concept on limited artificial datasets [22], we take advantage of the scalability of FHE to preserve the privacy of robust PRS models that contain a larger number

of significant SNPs. In addition to robust assessment with synthetic datasets to show how our model scales with increased SNP numbers, we apply our FHE-based protocol to a 110k-SNP PRS model for schizophrenia [23], which shows accuracy in predicting schizophrenia risk in a cohort of over 1200 individuals with minimal decrease in performance compared to non-encrypted PRS. We also thoroughly investigate the trade-offs for encryption parameters that influence computational accuracy, memory, and time for large PRS models, which are relevant for real-world healthcare applications. Our FHE protocol can preserve PRS while executing in a reasonable time; in fact, assessing one individual's PRS score with the 110k SNP model required just 12 seconds and 3.4G of memory with optimal parameters, while calculations for 1000 genotypes required five minutes with 113.8G memory. By using FHE to obtain meaningful results from robust PRS models while maintaining the security of the underlying genomic datasets, our study provides valuable insights into the potential use of FHE in genomics and healthcare. An implementation of our FHE protocol is available as an open-source software package called HEPRS at https://github.com/jiaqi54/HEPRS.

3 Results

3.1 Method Overview

Our work establishes a novel privacy-preserving protocol utilizing homomorphic encryption to predict phenotype risk from genomic data by involving three independent parties: the client, the modeler, and the evaluator (Fig. 1). Our protocol delineates distinct roles—clinicians could be clients providing genomic data, researchers or private companies as modelers creating the PRS, and hospitals or centralized systems as evaluators processing the encrypted genomes, which ensures effective use of genomic information while prioritizing patient confidentiality.

This protocol utilizes the CKKS encryption scheme [24], which is quantum secure and allows for a broader range of bootstrapped computations on encrypted data than other homomorphic encryption methods. Using both synthetic and real genomic datasets, we show that our encryption scheme efficiently calculates PRS values without sacrificing privacy or accuracy compared with traditional non-encrypted PRS methods. We further systematically analyze the various encryption parameters used in our method to report their effects on accuracy (R-squared and AUROC), memory usage, and computational time. Overall, our findings underscore the practicality of employing FHE in clinical genomics, particularly for PRS calculations where data privacy is paramount, without compromising computational integrity.

3.1.1 Evaluation Strategy

The FHE encryption framework involves the client, the modeler, and the evaluator [21] (Fig. 1). The roles within our protocol can be adapted to various configurations, aligning with the operational structures of clinical genomics environments. For instance, the client could be a clinician tasked with diagnosing or treating a patient, ensuring that the genomic data is utilized effectively while maintaining patient confidentiality. The modeler could represent a centralized data repository, akin to the UKBiobank resource, which manages PRS calculations and the underlying summary statistics for various disorders. The evaluator could be a hospital's biobank, which can securely access and process a patient's encrypted genome in conjunction with their electronic health record (EHR). In this case, the client has genomic data and wants phenotype predictions, the modeler trains a model for predictions, and the evaluator evaluates the model using the client's data and reports the forecast to the client.

Privacy is protected throughout the framework using homomorphic encryption, where the client's data and model are encrypted, and computations (i.e., calculating the PRS) are performed only in an encrypted form (Fig. 1). Specifically, the client first generates a public and secret key and encrypts the data using the public key. The client then shares the public key with the modeler and the evaluator, and the modeler encrypts their model with the public key. The client and modeler both share their encrypted data with the evaluator. The evaluator performs the computation using the encrypted model, encrypted data, and public key. Finally, the evaluator returns the encrypted predictions to the client, who then decrypts the results with their secret key. We assume no cross-talk between the client and the evaluator. It is also pertinent to acknowledge that the modeler, in handling GWAS summary statistics, is a custodian of sensitive information, as minor allele frequencies could allow for identifying individuals in the study [25]. In this context, the modeler would trust that the evaluator does not share the encrypted model with the client. This way, all data is encrypted, and only the client can access the plaintext predictions (Fig. 1).

[Figure 1]

3.2 Technical Overview of Key Aspects of FHE for PRS

We use the CKKS encryption scheme, implemented in the lattigo library [24], as the FHE method employed in our encrypted PRS calculation framework. The CKKS scheme gains NP-hard security from leveraging the Ring Learning With Errors (RLWE) problem. Homomorphic evaluation allows the scheme to perform a specific set of operations on ciphertexts, such as addition and multiplication, and produce a new ciphertext as a result. Given any operation that can be represented as a sequence of basic operations, the CKKS scheme repeatedly implements bootstrapping to apply homomorphic evaluation. The CKKS scheme is classified as lattice-based cryptography and is quantum secure [24]

Figure 2 outlines each important parameter and its meaning in the context of the encrypted PRS pipeline. We first encode both model β and genome X_i for J individuals using the CKKS plaintext encoding scheme. CKKS encodes vectors into polynomials of a degree defined by N with coefficients defined in an integer number ring described in more detail in the methods.

[Figure 2]

Figure 3 outlines how we encode the genomic vector into the encrypted domain to prepare for FHE evaluation. During encoding, we break each individual's genome vector X_i with dimension $1 \times M$ into a matrix of dimension $K \times N/2$. This becomes matrix $\boldsymbol{X_{e,i}}$ with dimension $K \times N/2$. After doing this process for each individual, we generate a three-dimensional tensor of dimension $J \times K \times N/2$. We repeat this process for vector β to get matrix $\boldsymbol{\beta_e}$ with dimension K x $N/2$. We ultimately compute the calculation $X_i \times \beta = y_i$, where y_i is the scalar output from the PRS model. We outline our algorithm in Figure 4 with a greater explanation of the individual steps in the methods section.

[Figure 3] [Figure 4]

In Figure 5, we highlight the important parameters that customize the CKKS encryption scheme for optimal results for a given task. The parameter N in the context of CKKS (and other homomorphic encryption schemes) represents the degree of the polynomial ring over which the encryption operates. CKKS encodes vectors into polynomials of a degree defined by N (we provide further explanations in the methods section). The choice of N determines the security and efficiency (larger values of N create a higher level of security but require more computational resources). Lastly, the value of N is typically chosen to be a power of 2 for efficient implementation, mainly because Fast Fourier Transform (FFT) algorithms, which are used for polynomial multiplication, are most efficient when operating on sequence lengths that have a power of 2. The value of N will determine

the optimal values of Q and P to maximize these tradeoffs. The Lattigo library provides optimal configurations for both these parameters. The genomic vectors are converted into a matrix when encoded into the encrypted domain defined by a cyclotomic polynomial, whose domain is defined by Q and N. Q determines the precision of encrypted genetic data, ensuring accurate PRS computation without loss of data integrity. P is critical during the relinearization step post-multiplication to safeguard the computation's accuracy. We describe each parameter in Figure 5.

[Figure 5]

3.3 Implementation Results

We evaluate the performance of our FHE-based PRS calculation method using both synthetic datasets and a real dataset for schizophrenia risk. We first applied our method to a synthetic dataset created using Hapgen2 [26], which contains 2,000 individuals and a range of SNPs for three artificial phenotypes (refered to as phenotypes 0, 1, and 2). We applied our method to a dataset of over 1,100 individuals from the PsychENCODE Consortium [27] evaluated against a 110K SNP PRS model for schizophrenia [23]. We compare the performance of our encrypted PRS with the non-encrypted PRS in terms of score correlation (Pearson r) and mean squared error (MSE) in phenotypic variance explained by the scores.

3.3.1 Overview of PRS derivations

We used two complementary approaches to assess the performance of our homomorphic encryption-based PRS calculations. We first generated synthetic datasets using Hapgen2 [26], a simulation method that produces genotypes with realistic linkage disequilibrium patterns to evaluate our approach under various conditions further. We simulated datasets with varying numbers of SNPs (30k to 130k) and sample sizes (200 to 30,000 individuals) and calculated PRS for three artificial phenotypes with different levels of heritability using ridge regression. By assessing our FHE method on both real and synthetic data, we demonstrate its versatility and potential for secure and private PRS calculations in diverse scenarios. We further applied our method to a real-world dataset for schizophrenia, utilizing PRS derived from a recent Psychiatric Genomics Consortium GWAS of over 320,000 individuals [23]. Specifically, we calculated PRS for 1,146 individuals (493 cases and 653 controls) from the PsychENCODE Consortium [27], using a standardized data processing pipeline based on best practice guidelines [28] followed by LDPred2 for PRS calculation [29]. The resulting PRS model contained 110,258 SNPs and achieved a pseudo-R^2 of 0.2229 in predicting schizophrenia diagnosis.

[Figure 6]

For both approaches, we found that our FHE procedure introduces only a trivial amount of error compared to the inherent uncertainty of the PRS value as determined from non-encrypted approaches, despite the framework's computational and memory limitations with increasing data size. Specifically, the phenotype variability explained by PRS (R^2) from our homomorphic encryption framework closely matches that of the underlying PRS models for both synthetic and real data, indicating the explained variance in phenotype within the population remains consistent post-encryption. For example, we observe a small MSE ($<1.5 \times 10^{-8}$) when applying FHE with varying R^2 values to the synthetic datasets (Figure 6A). After ridge regression analysis on the encrypted data, the R^2 values obtained for the three artificial phenotypes were 0.189, 0.466, and 0.822, respectively (compared with the preset values of 0.3, 0.6, and 0.9). Similarly, using the 110K SNP dataset for schizophrenia PRS, our encrypted PRS values have an R^2 value of 0.2231 (with PRS, sex, age, and genotype PCs as covariates), compared with 0.2229 for the non-encrypted model

(Figure 6B). This represents a mean squared error (MSE) of 2.27×10^{-6} between the non-encrypted PRS and the encrypted PRS (for moduli 2^{13}). This minimal MSE demonstrates that the encrypted PRS data yields comparable R^2 values to those obtained without encryption. This suggests that our FHE method introduces only a small error that would not lead to changes in the derived PRS.

Additionally, for both synthetic and real datasets, we observe a decrease in MSE as we increase the polynomial modulus degree N of the encryption scheme from 2^{13} to 2^{16}, which suggests a reduction in computational error with higher moduli. For example, the MSE reduces to 1.42×10^{-6} for the real dataset with a modulus of 2^{16} (Figure 6B). This suggests that a higher modulus can mitigate computational errors introduced by the encryption. Furthermore, we show in Figure 7A that all calculation methods maintain a constant AUROC. In Figure 7B, we show a Pearson correlation with r >.999 between the encrypted and non-encrypted schizophrenia PRS regardless of modulus degree.

3.4 Effects of Input Data Size on Runtime and Memory for Encrypted PRS

While several parameters may affect runtime and memory usage, our method can quickly and accurately calculate PRS using encrypted genotypes and PRS models, even when implemented on a personal laptop. For instance, using a MacBook Pro (M1, 2020), our method required about 10 seconds and 3 GiB of memory to calculate PRS for a specific sample using the 110K schizophrenia PRS model (modulus 2^{13}). Calculating PRS for a larger cohort of 1146 samples on a single CPU (Intel 6234) required 6 minutes and about 65 GB of memory when using the 110K SNP model with optimal parameters (Figure 6B). The run time for 2,000 samples in the 10K SNP synthetic dataset takes about 2 minutes for any of the three synthetic phenotypes (Figure 6A).

Figure 6B demonstrates that the protocol utilizing homomorphic encryption (HE) for dot product calculation is roughly three orders of magnitude slower than plaintext calculations (on a single-CPU node - Intel 6234). When we do the dot product calculation in Golang, it takes 0.303 seconds; however, a more common way to calculate the PRS is with LDPred2 in R, which only takes 0.073 seconds. If we compare the dot product within Golang, the HE calculation (187.8s) takes around 623 times longer. Compared to standard ways of calculating the HE PRS, the calculation takes 2572 times longer. This contrast highlights the computational burden introduced by encryption protocols and the possible time added by the Golang language.

[Figure 8]

Using a synthetic dataset allowed us to vary both the input sample size and number of SNPs of the underlying PRS model to determine their effects on the performance of our method. We first assessed the influence of the number of SNPs on the time taken to calculate PRS using FHE for 50 samples (Fig. 8). We observe a direct correlation between the increase in SNP numbers—from 10,000 to 130,000—and the computational time required–from 5 seconds to 36 seconds, demonstrating approximately a linear relationship under the tested conditions (Fig. 8A). Fig. 8B expands on this by showing that memory requirements also rise in a similar linear fashion (from 0.8 GB to 9 GB) with the increase in SNP count, which is a critical consideration for the practical application of FHE in clinical settings.

Next, we varied the number of individuals for which we performed the encrypted PRS calculations, using a constant genome size of 130K SNPs and predicting for artificial phenotype 0. Similar to the number of input SNPs, we found that the number of individuals included in a batch has a linear relationship with the time to completion (Fig. 8C), ranging from about 1 minute for 200 individuals to 11 minutes for 2000 individuals. The required memory also positively correlates with the batch size, from 17 GB for 200 individuals to 114 GB for 2000 individuals (Fig. 8D). These data point to a predictable increase in analysis time with larger cohorts, reinforcing the scalability

of our FHE protocol for PRS calculation in real-world scenarios with substantial genomic datasets.

3.5 Encryption Parameters affect Accuracy and Runtime for PRS Calculation

We showed above that increasing the degree N of the polynomial, or ring modulus, parameter of the CKKS encryption scheme will decrease the errors introduced in the encrypted calculations. However, increasing the polynomial modulus will also increase the model runtime.

[Figure 9]

From the results obtained with the 110K SNP schizophrenia PRS model, we observed that the time required for the encrypted model to generate PRS predictions increased with the polynomial modulus (Fig. 9A). Evaluating PRS for 1,146 individuals using the 110K SNP model in plaintext (infinitesimal mode of LDPred2 for only the evaluation part) takes typically 0.07 seconds on a single CPU (Intel 6234), while for FHE, even using the smallest modulus degree 2^{13} takes about 6 min, and the time further increases to 20 min for the modulus degree 2^{16}. When breaking down the time cost into different stages of the encryption calculation, we find that most of the time is spent after the encryption step and during the model operation phase, corresponding to the actual computation of the encrypted PRS (Fig. 9B).

Fig. 9C presents the memory allocation for running the program across different ring modules. Contrary to what we expected, the memory usage remain largely invariant through repeated experiments, ranging from 60 GB to 80 GB, which only introduces a moderate increase in memory usage compared to the plaintext model, which is 50 GB. Thus, our FHE method for PRS calculation allows end users to account for the trade-off between protocol time and error mitigation according to their needs by customizing the polynomial modulus parameter.

4 Discussion

In this work, we implemented a FHE-based method for accurate and efficient PRS calculations without compromising patient privacy. Here, we show that the marginal difference of error introduced by the encryption scheme does not affect the overall error of the PRS model. We also show that different parameters affect the accuracy of results and the trade-off between accuracy and time within the FHE framework. Although FHE substantially increases the computational burden for PRS calculation compared to plaintext models, roughly three orders of magnitude slower, the cost remains within a reasonable range for practical applications. These findings suggest that our HE-based method is viable for securely computing PRS while maintaining a balance between privacy, accuracy, and computational efficiency.

The bulk of our PRS calculation is done within our function `HE_inner_product()`. While performing this operation is a simple calculation, the nature of large genomic data requires some reworking of the encrypted data storage and sequence of computation to arrive at an accurate phenotype prediction without using too much time or storage. In the methods section, we provide an intuitive explanation of how we utilize Ring Learning With Error to compute the dot product.

As explained in the methods, the modeler must share the encrypted model with the evaluator; if the evaluator shared this encrypted model with the client, the client could decrypt the model without the modeler's consent and thus break privacy requirements. The assumption that the evaluator and client never collude in our encryption schema is reasonable if we consider the evaluator to be a healthcare entity (e.g., a hospital or healthcare insurance institution). These organizations might be associated with a research group that owns the model. We assume the evaluator would want to keep a good relationship with the modelers and not erode trust. However, the trust between the modeler and evaluator poses a framework vulnerability.

About four to seven percent of an average healthcare system's IT budget is spent on cybersecurity. This means that from 2020 to 2025, healthcare organizations will allocate $125B to cybersecurity [30]. While hospitals use a large portion of this budget to defend the Internet of Things (IoT), some of this capital is spent protecting healthcare data, which increasingly includes patient genomic data. One could envision this protocol significantly reducing vulnerabilities if patient genomes were stored in encrypted formats and only the keys to decrypt this data were maintained under high security. Such an encryption framework would reduce vulnerabilities as only encrypted genomic data would be transferred instead of raw genomic data. However, privacy would be breached when the evaluator colludes with the client and reveals the underlying PRS weights.

In this work, we show the possibility and practicality of generating PRS model predictions without revealing a genome. We demonstrate that memory requirements increase with the increase in accuracy determined by specified parameters and how time scales roughly linearly with SNPs included in the model. Lastly, we show that the error introduced to apply homomorphic encryption is negligent compared to the inherent error of the model itself.

5 Conclusion

Considering the recent advances in FHE, we view PRS as an exciting new avenue for applying this technology. The strategic use of genomic information has the potential to significantly contribute to personalized healthcare, aiming to enhance health outcomes and elevate care standards. The necessity for patients to undergo genomic sequencing is paramount for the realization of this technology's benefits. Concurrently, the escalation of genetic data mandates stringent security and privacy measures. The challenges of applying analytical models to decipher relevant information from the vast amounts of data necessary to encode a genome are non-trivial. Our research offers a solution that addresses these multifaceted issues. While our focus is on PRS, the principles of FHE can potentially be applied to a spectrum of genomic computations. The generalizability of FHE to other areas in clinical genomics, such as variant annotation, genotype imputation, and more complex predictive modeling, warrants further investigation. This research paper takes an initial step toward exploring such applications.

6 Methods

6.1 CKKS Encryption Scheme

We use the CKKS encryption scheme, implemented in the lattigo library [24], as the FHE method employed in our encrypted PRS calculation framework. The CKKS scheme gains NP-hard security from employing the Ring Learning With Errors (RLWE) problem.

6.2 Key Generation

The key generation step is implemented using the Encrypt_input function in the lattigo library. In this function, we employ the CKKS parameters to create a new key generator through the "NewKeyGenerator" method provided in the CKKS library within lattigo. The "GenKeyPair" method generates a public-private key pair (pk and sk), and the "GenRelinearizationKey" method creates the relinearization key that allows efficient computation on encrypted data. This method requires the secret key and max_level parameters, the latter of which we set to 2 (for addition and subtraction calculations). Finally, the "GenRotationKeysForInnerSum" method generates rotation keys, which allow the operations to involve rotations of encrypted data (for instance, in the inner

product calculation). The function returns the generated keys and parameters for further use in the encryption and evaluation steps of the homomorphic encryption scheme.

6.3 Implementation of Homomorphic Encryption

First, we load and encrypt the genotype data on the client side. During this step, the client generates the secret decryption key, which generates the public encryption key, public relinearization key, and public rotation key. The public keys are shared with all parties. The client next encrypts the genotype data to transform each individual's genotype into a 2-dimensional ciphertext matrix. The chosen parameters (in our polynomial modulus investigation) determine the vector's dimensions.

Next, the modeler loads and encrypts the PRS model using the public encryption key provided by the client. A similar scheme, dependent on the parameters, saves each model as a two-dimensional ciphertext matrix.

After this, the evaluator can load and evaluate the encrypted model on the encrypted genomes. In this arrangement, we assume that the evaluator and client are trustworthy and cannot collude; the encrypted model never gets shared with the client. The evaluator then uses the public relinearization and rotation keys (provided by the client) and the encrypted model and genome to perform a polynomial evaluation to arrive at the encrypted phenotype (i.e., PRS value). This step implements an inner product calculation performed using homomorphic encryption, as detailed below.

Finally, once the client has received the encrypted phenotype result (risk score for disease), the client can decrypt the result with the secret key.

We implement the protocol using Golang. Go's suitability for cryptography is underscored by its comprehensive standard library that includes support for common cryptographic algorithms alongside features such as concurrency support, inherent memory safety, and strong typing with compile-time checks.

6.4 Implementation of Inner Product using Homomorphic Encryption

We first encode both model β and X_i genome for J individuals using the CKKs plaintext encoding scheme. During encoding, the integer-based plaintext vector X_i (representing genotype values of 0, 1, or 2) is encoded into the domain $\frac{Z_Q[X]}{X^N+1}$. This requires us to encode our vectors into a polynomial where all coefficients are $\in Z_Q$ and all operations are performed as mod $X^N + 1$.

[Figure 2,3,4]

For the next section, we focus on the calculation of a single individual's PRS. Here, the encrypted genomic data matrix $\boldsymbol{X_{e,i}}$ encapsulates the genotypes, while the model matrix $\boldsymbol{\beta_e}$ holds the beta values. Both matrices have dimensions $K \times N/2$, aligning with the encryption parameters outlined in Figure 3.

Algorithm 1, as depicted in Figure 4, is instrumental in the calculation of PRS by implementing the dot product of encrypted vectors. Within the for-loop, each row vector from the genomic data matrix $\boldsymbol{X_{e,i}}$ is element-wise multiplied with the corresponding row in the model matrix β_e using the MulRelinNew() function from Lattigo[20]. This multiplication yields new vectors, $v_h = \boldsymbol{X_{e,i}}[\mathrm{h}]*\boldsymbol{\beta_e}[\mathrm{h}]$, representing the weighted contribution of each genotype to the PRS, here * represents an element-wise multiplication. The resulting vector aggregates these products, accumulating the combined effect of genotypes and beta values on the PRS. The Lattigo InnerSumLog() function subsequently sums the polynomial terms within the result vector, reflecting the final summation of individual genetic risk factors. This process effectively translates the encrypted genomic information and model predictions into a single encrypted PRS value for each individual. This function fails

to perform some specialized processes currently executed by more advanced calculation methods such as the plink2 clump function [31]; however, this process calculates a sufficient PRS that differs from the Plink 2 model by 0.000129%.

6.5 Synthetic Data Generation and Ridge Regression

Hapgen2 [26] is a simulation method that resamples known haplotypes quickly and efficiently to produce samples with linkage disequilibrium (LD) patterns that mimic those in real data. Hapgen2 is based on the Li and Stephens (LS) model, [32], of LD. The method takes in a reference panel of haplotypes as input and generates genotype encodings based on the haplotypes. We used Hapgen2 to simulate genotype encodings for 40,000 individuals across a varying number of SNP sizes, (30k, 50k, 100k, and 130k), based on the 1000 Genomes project reference panel. We later split these individuals into training groups and testing groups for ridge regression of sizes 30,000 and 2,000, respectively, for each SNP size.

We consider a high-dimensional regression framework for polygenic modeling and prediction

$$Y = X\beta + \epsilon \tag{1}$$

where Y is an $N \times 1$ vector of phenotype values with $Y_i \in \mathbb{R}$ and N being the number of individuals in the dataset. The parameter $X^{N \times M}$ is the genotype encoding matrix with $X_{im} \in \{0, 1, 2\}$ and M being the number of SNPs in the genotype encoding. The corresponding effect size of each SNP m is β_m. The parameter ϵ is a vector of residual effects for each individual. We are interested in estimating β. We will assign a Gaussian prior on each β_m.

$$\beta_j \sim N(0, \sqrt{h^2/(Mp)}) \tag{2}$$

,

where p is the percentage of SNPs with non-null effect size, and h^2 is the total SNP-heritability. Hence, we assume that each SNP with a non-zero effect size explains an equal portion of the total SNP-heritability. To model the impact of SNPs with zero effect size, we select $M(1-p)$ random SNP effect size locations in the β vector and set the value at those indexes equal to zero. Each of the three artificial phenotypes were generated under the assumption that approximately 90% of the SNPs had non-zero effect size, or $p = 0.9$, a level of sparsity generally consistent with realistic human data [33]. The three levels of SNP heritability used were $h^2 = 0.3, 0.6, 0.9$.

We used multiple SNP sizes, (30k, 50k, 100k, and 130k), for our simulation data to show that the FHE scheme would apply well to real-world settings where the SNP size varies depending on which illnesses are being studied. Both values of p and h^2 were fixed across the training population and the testing population to maintain consistency across the two data populations. The population count of the training dataset is $N = 30,000$, and for the testing data the count is $N = 2000$.

Recall that the estimator of Ridge Regression $\hat{\beta}$ is equal to the mean of the posterior distribution under a Gaussian prior. Hence, the choice of a Gaussian prior on β in this PRS setting naturally leads us to use Ridge Regression. Moreover, an extra $1's$ column was added for the intercept term. We note that ridge regression is one of many models researchers use for calculating PRS. More advanced models, such as Bayesian hierarchical models using Gibbs sampling or variational inference, may be more accurate because they have relaxed assumptions on the parameters of the training data. Although we sacrifice computational accuracy in using ridge regression, we gain faster time complexity. In real-world settings, this trade-off is not as important since the model is built once off of training data and not repeatedly calculated. It is important to note that the sacrifice on accuracy is made in the model selection and not in the FHE scheme. Hence, the FHE

scheme's performance should be judged purely on the accuracy of the decrypted prediction values compared to the ciphertext prediction values.

6.6 Schizophrenia PRS generation

We calculated schizophrenia PRS for 1,146 individuals from the PsychENCODE Consortium study, including 493 individuals diagnosed with schizophrenia and 653 controls [27]. Imputed genotypes and metadata for these individuals were downloaded from the PsychENCODE Consortium data portal [27], with more than 4.3M imputed SNPs available for each individual. Schizophrenia GWAS summary statistics were selected from a recent Psychiatric Genomics Consortium study of more than 320,000 individuals, and were downloaded from the PGC portal [23]. Using these inputs, we implemented a standardized data processing pipeline for calculating PRS based on best practice guidelines [28]. We first performed quality control filtering on the GWAS summary statistics by removing SNPs with INFO scores less than 0.8 as well as ambiguous and duplicate SNPs. Next, after lifting over coordinates and fixing alleles of the PsychENCODE genotypes to the hg38 reference genome, we performed strand-flipping using the snpflip software package [34]. We removed SNPs with minor allele frequency ¡0.05 or Hardy-Weinberg equilibrium $p < 1 \times 10^{-6}$ (resulting in more than 2.6M SNPs per sample) and removed individuals with ¿3SD genotype heterozygosity rate (F coefficient, calculated using SNPs in 200kb windows with linkage disequilibrium [LD] $r^2 > 0.25$). An additional strand-flipping step was performed using snpflip to match alleles between the sample genotypes and summary statistic SNPs. Finally, six genotype principal components (PCs) were calculated for each sample using the –pca command in PLINK2 [35].

We used the LDpred2 function within the bigsnpr software package to calculate PRS scores for each individual [29]. After filtering for SNPs present in the HapMap3 dataset, we calculated LD Scores for 110,258 matching SNPs between the PsychENCODE genotypes and summary statistics using the snp_cor function and centimorgan map units from the 1000 Genomes project. We then regressed the LD scores against the log-scaled summary beta values using the snp_ldsc function and calculated a baseline heritability estimate based on the output (h^2_est, here equal to 0.1270). Using these modified summary statistics, we then calculated PRS for each individual using the infinitesimal and grid models of LDpred2. For the grid model, we assessed combinations of (a) 17 input SNP p-value filters ranging from 1.0×10^{-4} to 1.0, (b) three potential values of h^2 (0.7, 1.0, and $1.4\times h^2_est$), and (c) spare or non-sparse grid models, for a total of 102 PRS outputs. (We note that the LDpred2-auto model did not converge for our datasets, and thus was not used in our analysis.) To assess the performance of each PRS model, we generated logistic regression models for PRS scores towards the binary schizophrenia diagnosis of each individual, with sex, age, and six genotype PCs included as covariates. We then calculated Nagelkerke $pseudo\text{-}R^2$ values for each regression model, which were compared against those for a null logistic regression model consisting solely of the sex, age, and genotype PC covariates.

Overall, we found that both the infinitesimal model ($pseudo\text{-}R^2$=0.2008) and best-performing grid model ($pseudo\text{-}R^2$=0.2229; at p=0.32, $h^2=0.7\times h^2_est$, and sparse grid model) performed substantially better than the null model ($pseudo\text{-}R^2$=0.0809). We selected the PRS outputs derived from the best-performing grid model for downstream analysis. Correlations and AUROC calculations were performed in Python using the sklearn library.

In contrast to the LDpred2 model, our PRS calculation approach used for the Homomorphic Encryption protocol, conducts SNP filtering as a preprocessing measure. In this way, we ensure that the input to our model consists of already filtered SNPs. Additionally, our model does not explicitly manage linkage disequilibrium (LD) in the same detailed manner. Instead, it operates similarly to a straightforward PRS model, which assumes that the selected SNPs are representative

of all SNPs that contribute to a phenotype and have been preprocessed to mitigate LD concerns.

References

[1] W Richard McCombie, John D McPherson, and Elaine R Mardis. "Next-Generation Sequencing Technologies". In: *Cold Spring Harbor Perspectives in Medicine* 9.11 (Nov. 2018), a036798. DOI: 10.1101/cshperspect.a036798. URL: https://doi.org/10.1101%2Fcshperspect.a036798.

[2] Emil Uffelmann et al. "Genome-wide association studies". In: *Nature Reviews Methods Primers* 1.1 (Aug. 2021). DOI: 10.1038/s43586-021-00056-9. URL: https://doi.org/10.1038%2Fs43586-021-00056-9.

[3] Teri A. Manolio. "Bringing genome-wide association findings into clinical use". In: *Nature Reviews Genetics* 14.8 (2013), pp. 549–558. DOI: 10.1038/nrg3523. eprint: 2013Jul9.

[4] Chris Finan et al. "The druggable genome and support for target identification and validation in drug development". In: *Science Translational Medicine* 9.383 (2017), eaag1166. DOI: 10.1126/scitranslmed.aag1166. eprint: 2017Mar29.

[5] Cathryn M. Lewis and Evangelos Vassos. "Polygenic risk scores: from research tools to clinical instruments". In: *Genome Medicine* 12.1 (2020), p. 44. DOI: 10.1186/s13073-020-00742-5. eprint: 2020May18.

[6] Leo P. Sugrue and Rahul S. Desikan. "What Are Polygenic Scores and Why Are They Important?" In: *JAMA* 321.18 (May 2019), pp. 1820–1821. DOI: 10.1001/jama.2019.3893.

[7] Nicholas A. Marston, James P. Pirruccello, Giorgio E. M. Melloni, et al. "Predictive Utility of a Coronary Artery Disease Polygenic Risk Score in Primary Prevention". In: *JAMA Cardiology* 8.2 (2023), pp. 130–137. DOI: 10.1001/jamacardio.2022.4466.

[8] Nasim Mavaddat et al. "Polygenic Risk Scores for Prediction of Breast Cancer and Breast Cancer Subtypes". In: *American Journal of Human Genetics* 104.1 (2019), pp. 21–34. DOI: 10.1016/j.ajhg.2018.11.002. eprint: 2018Dec13.

[9] Loïc Yengo and et al. "A saturated map of common genetic variants associated with human height". In: *Nature* 610 (7933 2022), pp. 704–712. DOI: 10.1038/s41586-022-05275-y.

[10] Haoyu Zhang and et al. "Genome-wide association study identifies 32 novel breast cancer susceptibility loci from overall and subtype-specific analyses". In: *Nature Genetics* 52 (6 2020), pp. 572–581. DOI: 10.1038/s41588-020-0609-2.

[11] Xinghua Shi and Xintao Wu. "An overview of human genetic privacy". In: *Annals of the New York Academy of Sciences* 1387.1 (Jan. 2017), pp. 61–72. ISSN: 0077-8923. DOI: 10.1111/nyas.13211.

[12] C. Bycroft et al. "The UK Biobank resource with deep phenotyping and genomic data". In: *Nature* 562 (2018), pp. 203–209. DOI: 10.1038/s41586-018-0579-z.

[13] Karl Freund. "IBM Announces Two Innovations To Advance Quantum Computing". In: *Forbes* (2022). URL: https://www.forbes.com/sites/karlfreund/2022/11/10/ibm-announces-two-innovations-to-advance-quantum-computing/?sh=5d2fa5421b60.

[14] Yue Wang, Cheng Si, and Xintao Wu. "Regression model fitting under differential privacy and model inversion attack". In: *Proceedings of the 24th International Conference on Artificial Intelligence*. IJCAI'15. Buenos Aires, Argentina: AAAI Press, 2015, pp. 1003–1009. ISBN: 9781577357384.

[15] Aaruni Kaushik. *Investigating Resilience of Levelled Fully Homomorphic Encryption System Against Data Scientific Attacks*. Master's thesis. Available at https://akaushik.edufor.me/resources/MasterThesis.pdf. Erwin-Schrödinger-Straße 52, 67663 Kaiserslautern, Germany, Feb. 2023.

[16] Sarezh Paevl Nyland. *Implemention and Evaluation of a Private Set Intersection Protocol Built With Fully Homomorphic Encryption*. Master's thesis. Available at https://uis.brage.unit.no/uis-xmlui/handle/11250/3089847. Kjell Arholms gate 41, 4021 Stavanger, Norway, Sept. 2023.

[17] Craig Gentry. "Computing arbitrary functions of encrypted data". In: *Communications of the ACM* 53.3 (Mar. 2010), pp. 97–105. ISSN: 0001-0782. DOI: 10.1145/1666420.1666444.

[18] Mostefa Kara et al. "A fully homomorphic encryption based on magic number fragmentation and El-Gamal encryption: Smart healthcare use case". In: *Expert Systems* 39.5 (2022), e12767. DOI: https://doi.org/10.1111/exsy.12767. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/exsy.12767. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/exsy.12767.

[19] Jean Louis Raisaro et al. "Protecting Privacy and Security of Genomic Data in i2b2 with Homomorphic Encryption and Differential Privacy". In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 15.5 (Sept. 2018), pp. 1413–1426. DOI: 10.1109/TCBB.2018.2854782.

[20] *Lattigo v5*. Online: https://github.com/tuneinsight/lattigo. EPFL-LDS, Tune Insight SA. Nov. 2023.

[21] *Human Genome Privacy Challenge 2022*. Accessed: September 2022. 2022. URL: http://www.humangenomeprivacy.org/2022/.

[22] Shaedul Islam, Hüseyin Demirci, and Gabriele Lenzini. "Privacy-preserving Polygenic Risk Scoring using Homomorphic Encryption". In: *Proceedings of the 18th Scandinavian Conference on Health Informatics*. Aug. 2022, pp. 124–127. DOI: 10.3384/ecp187021. URL: https://doi.org/10.3384/ecp187021.

[23] Vassily Trubetskoy et al. "Mapping genomic loci implicates genes and synaptic biology in schizophrenia". In: *Nature* 604 (2022), pp. 502–508. DOI: 10.1038/s41586-022-04434-5. URL: https://doi.org/10.1038/s41586-022-04434-5.

[24] Andrey Kim, Antonis Papadimitriou, and Yuriy Polyakov. *Approximate Homomorphic Encryption with Reduced Approximation Error*. Online: https://eprint.iacr.org/2020/1118.pdf. New Jersey Institute of Technology, Samsung Advanced Institute of Technology, Duality Technologies. Dec. 2021.

[25] David W Craig et al. "Assessing and managing risk when sharing aggregate genetic variant data". In: *Nature Reviews Genetics* 12.10 (Sept. 2011). Erratum in: Nat Rev Genet. 2011 Nov;12(11):801, pp. 730–736. DOI: 10.1038/nrg3067. eprint: https://doi.org/10.1038/nrg3067.

[26] Zhan Su, Jonathan Marchini, and Peter Donnelly. "HAPGEN2: simulation of multiple disease SNPs". In: *Bioinformatics* 27 16 (2011), pp. 2304–5.

[27] Daifeng Wang and et al. "Title of the Paper". In: *Science* 362 (6420 2018), eaat8464. DOI: 10.1126/science.aat8464.

[28] Shing Wan Choi, Tim Shuk Han Mak, and Paul F O'Reilly. "Tutorial: a guide to performing polygenic risk score analyses". In: *Nat Protoc* 15.9 (Sept. 2020). Epub 2020 Jul 24, pp. 2759–2772. DOI: 10.1038/s41596-020-0353-1.

[29] Florian Prive, June Arbel, and Bjarni J. Vilhjálmsson. "LDpred2: better, faster, stronger". In: *Bioinformatics* 36.22-23 (Apr. 2021), pp. 5424–5431. DOI: 10.1093/bioinformatics/btaa1029.

[30] Steve Morgan. *Healthcare industry to spend $125 billion on cybersecurity from 2020 to 2025*. Nov. 2020. URL: https://cybersecurityventures.com/healthcare-industry-to-spend-125-billion-on-cybersecurity-from-2020-to-2025.

[31] Christopher C Chang et al. "Second-generation PLINK: rising to the challenge of larger and richer datasets". In: *GigaScience* 4 (1 Feb. 2015). Published: 25 February 2015. DOI: 10.1186/s13742-015-0047-8.

[32] Na Li and Matthew Stephens. "Modelling Linkage Disequilibrium , And Identifying Recombination Hotspots Using SNP Data". In: 2003.

[33] Huwen Shi et al. "Localizing Components of Shared Transethnic Genetic Architecture of Complex Traits from GWAS Summary Data". In: *American Journal of Human Genetics* 106.6 (June 2020), pp. 805–817. DOI: 10.1016/j.ajhg.2020.04.012. eprint: https://doi.org/10.1016/j.ajhg.2020.04.012.

[34] Endre Bakken Stovner and Brian Sebastian Cole. *snpflip*. https://github.com/biocore-ntnu/snpflip. 2019.

[35] Christopher C Chang et al. "Second-generation PLINK: rising to the challenge of larger and richer datasets". In: *GigaScience* 4.1 (Dec. 2015), s13742-015-0047–8. DOI: 10.1186/s13742-015-0047-8.

List of abbreviations

FHE Fully Homomorphic Encryption

PRS Polygenic Risk Scores

CKKs Cheon-Kim-Kim-Song

SNP Single Nucleotide Polymorphism

GWAS Genome-Wide Association Studies

PGC Psychiatric Genomics Consortium

MSE Mean Squared Error

RLWE Ring Learning With Errors

LDPred2 LDPred version 2

PC Principal Component

EHR Electronic Health Record

IoT Internet of Things

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

Not applicable.

Availability of data and materials

The datasets generated and/or analyzed during the current study are available from the corresponding author upon reasonable request. PsychENCODE genotypes are available for download by approved researchers at http://resource.psychencode.org/, and GWAS summary statistics for schizophrenia are available from the Psychiatric Genomic Consortium portal at https://pgc.unc.edu/for-researchers/download-results. Code is available as a GO package at https://github.com/jiaqi54/HEPRS, along with a small example dataset that includes synthetic genotypes and phenotypes generated using Hapgen2 for tutorial purposes.

Competing interests

The authors declare that they have no competing interests.

Funding

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.

Authors' contributions

Acknowledgments

Authors' information

Figure1:

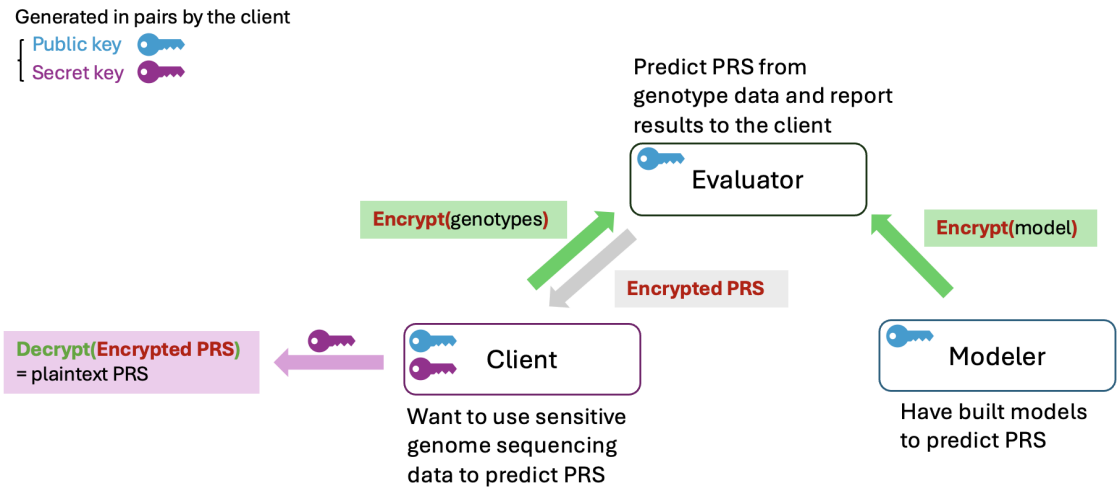

Figure 1: **Schematic illustration of the encryption setting**. The public key and secret key are generated in pairs by the client. The public key is shared with all three parties, while the secret key is only known to the client. The client performs homomorphic encryption of their sensitive genetic data using the public key, and the modeler encrypts their model also using the same public key. The clients' genomic data and the model are sent to the evaluator in their encrypted form. The evaluator cannot decrypt the data or the model without the secret key. However, the evaluator is able to evaluate the genetic data and obtain model predictions, which remain encrypted. The encrypted phenotype predictions are reported back to the client, who can decrypt the data using the secret key, and read predictions.

Figure 2: Parameters for Homomorphic Encryption

Symbol	Description	Dimension
N	determines length of encrypted vector	scalar
slots	maximum vector size	N/2
M	number of SNPs included in PRS model	scalar
K	number of encryption vectors per one genome	scalar
X_i	genomic data vector	$M)$
$\boldsymbol{X_{e,i}}$	encrypted genomic matrix	$(K, N/2)$
β	unencrypted PRS model vector	(M)
$\boldsymbol{\beta_e}$	encrypted PRS model matrix	$(K, N/2)$
J	number of individuals	variable
$Y_{e,i}$	encrypted result vector	$(N/2)$
y_i	unencrypted result	scalar

Figure 2:

N is the polynomial degree modulus which determines how many values can be encoded in one batch. A maximum of N/2 values can be encoded at once. Slots is this number, in other words the maximum vector size for the given polynomial modulus. M is the number of snps included in the polygenic risk score. J is the number of individuals that we tested. K is the number of encrypted vectors that make up one encrypted genome($1 + \frac{M-1}{slots}$) . X_i is the genomic vector for individual i, and $\boldsymbol{X_{e,i}}$ is the encrypted genomic matrix for invidiaul i. β is the vector containing model weights for the Polygenic Risk Score and $\boldsymbol{\beta_e}$ is the model matrix with weights encrypted. $Y_{e,i}$ is the encrypted results vector and y_i is the result scalar unencrypted that communicates an individuals risk for disease.

Figure 3: Transfer of genotype into the encrypted domain.

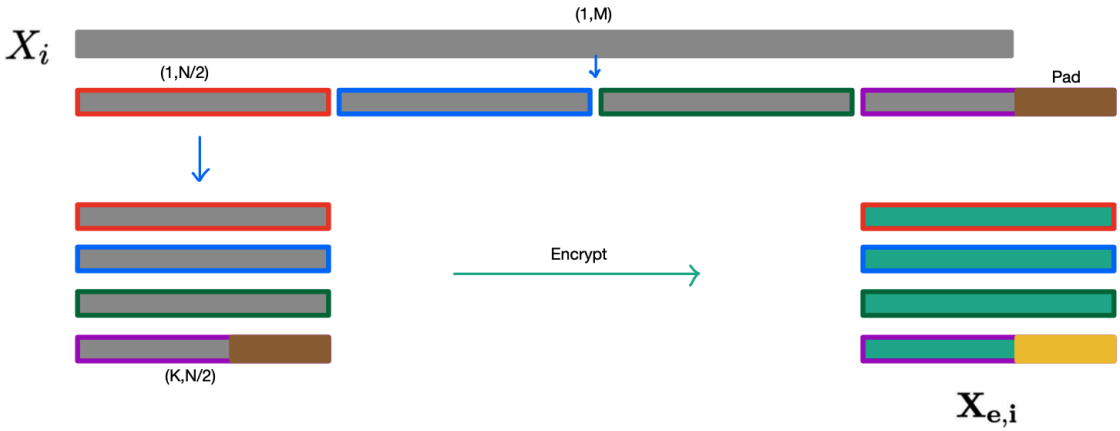

Figure 3 shows how we encode the genomic vector X_i, into a genomic matrix in the encrypted domain **X_e,i.**

Figure 4:

Algorithm 1 Multiply vectors and add together

Input: $\mathbf{X_{e,i}}$: fixed-length matrix ckks.Ciphertext $\mathbf{fi_e}$: fixed-length matrix ckks.Ciphertext slots: integer

```
result = vector(length = N/2)                    ▷ this initializes a new ciphertext vector
for h in 1:K do
    v_h = multiply X_{e,i}[h] with β_e[h] elementwise    ▷ len(v_h) = len(X_{e,i}[h])
    Y_{e,i} = Y_{e,i} + v_h                          ▷ addition step
end for
Y_{e,i} = polynomial rotational addition of (Y_{e,i})
return (Y_{e,i}[0])        ▷ return the first element of the result vector (complex scalar)
```

Caption: In algorithm 1, we show how to perform a dot product operation on encrypted data using functions within Lattigo's library.

Figure 5: Parameters Used in Lattigo

$\log_2 N$	13
Q	33+ 30 + 30 + 30 + 30 + 30
P	35
Default Scale	2^{30}

Figure 5 shows the parameters that are customizable for specific applications within the CKKs encryption scheme.

A

Phenotype	Modulus	R2 of plaintext model	R2 of encrypted model	MSE between plaintext and encrypted model
0	2^{13}		0.18892715	0.00000001397358
	2^{14}		0.18892446	0.0000000007777809
	2^{15}		0.18892441	0
	2^{16}	0.18892353	0.18892443	0
1	2^{13}		0.4664542	0.000000008085253
	2^{14}		0.46646012	0.0000000001919872
	2^{15}		0.46645956	0
	2^{16}	0.46645884	0.46645956	0
2	2^{13}		0.82261809	0.00000001167814
	2^{14}		0.82261557	0.000000001426231
	2^{15}		0.82261601	0
	2^{16}	0.8226163	0.822616	0

B

Modulus	R2 of LDPred2 model	R2 of encrypted model	R2 between encrypted and LDPred2 model predictions	MSE between LDPred2 and encrypted model	Time for LDPred2 Infinitesimal model calculation (s)	Time for HE calculation (s) [with reading and writing]	Time for HE calculation (s) [without reading and writing]	Time for plaintext calculation in Golang (s) [with reading and writing]	Time for plaintext calculation in Golang (s) [without reading and writing]
2^{13}		0.2230825	0.99999916	0.00000226829214		365.22 ± 9.48	187.8		
2^{14}		0.223064	1	0.00000143001642		507.9 ± 13.98	319.2		
2^{15}		0.223067	1	0.00000142534149		691.98 ± 30.78	470.4		
2^{16}	0.2226318	0.2230669	1	0.00000142533753	0.073	1274.28 ± 26.1	960	6.224	0.303

Figure 6: Tables comparing time and accuracy between PRS predicted by HE and non-encrypted models. (A) Comparison of PRS generated from plaintext ridge regression models and encrypted models. Data used here was generated using Hapgen2, including three phenotypes corresponding to different levels of SNP heritability ($h^2 = 0.3,\ 0.6,\ 0.9$), which we refer to as phenotype 0, 1 and 2. The R^2 between simulated phenotypes and PRS predicted by the plaintext and encrypted models are shown. MSE values are shown to compare the predictions with and without encryption. Statistics shown here were from a model with 10,000 SNPs and 2,000 individuals. (B) Comparison of the PRS generated from the encrypted model and LDpred2model. The coefficient of determination (R^2) for LDpred2and encrypted models are generated from fitting logistic regression models towards the binary schizophrenia diagnosis of each individual, with PRS, sex, age, and six genotype PCs included as covariates. The Nagelkerke *pseudo*-R^2 values for each regression model are reported. The R^2 between

encrypted and LDpred2 models are calculated from a linear regression between encrypted and unencrypted PRS. The mean squared error (MSE) is also calculated between encrypted and unencrypted predictions. The time for running our encrypted model is listed for each modulus specified. The input consists of 1146 individuals with 110,258 SNPs. Time reported for HE calculation is the mean ± standard deviation record from 10 independent runs. Times for HE, LDPred2, and plaintext Golang (with and without reading and writing) calculations are all performed using a single 6234 CPU.

Figure 7:
Figure 7A:

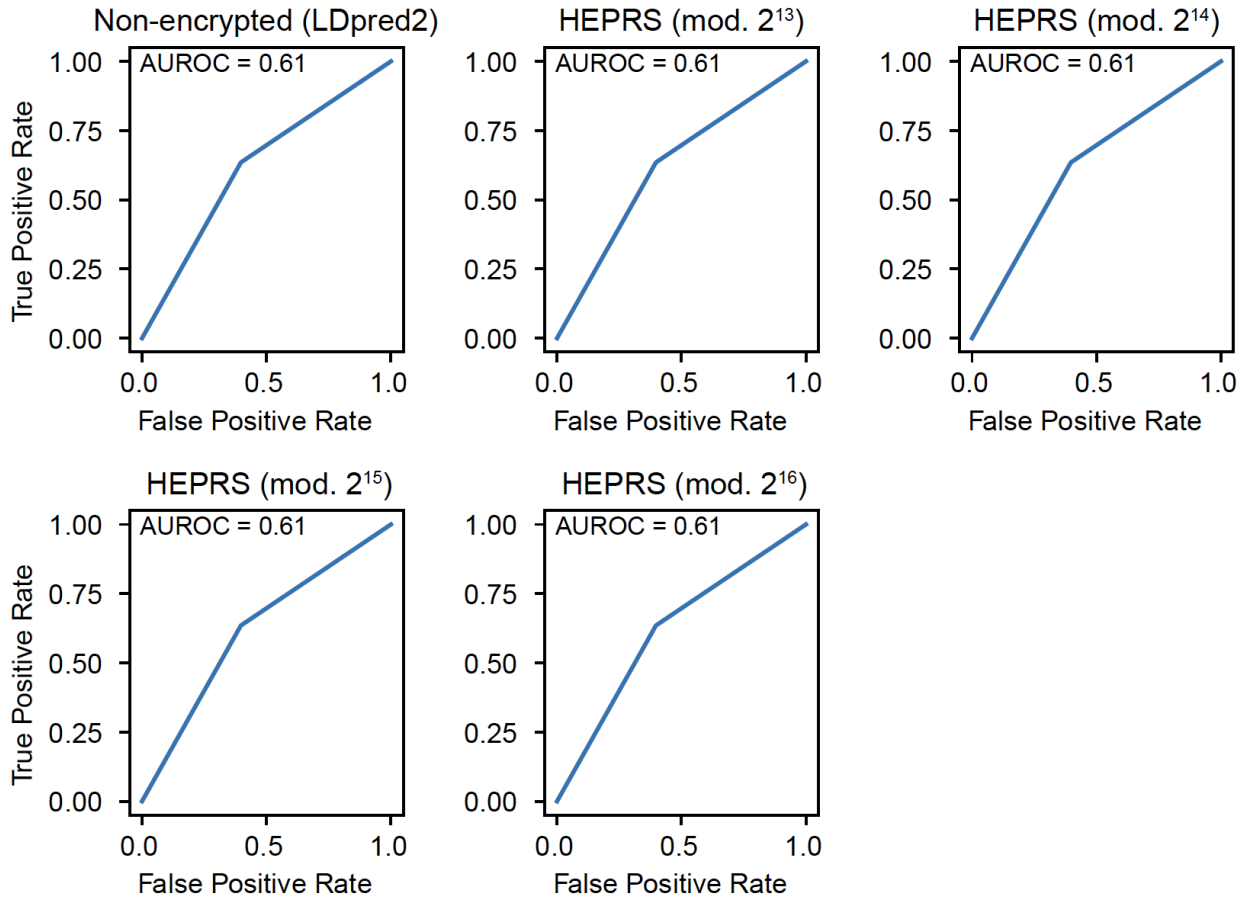

Figure 7A: Receiver operating characteristic (ROC) curves for prediction of schizophrenia phenotypes based on non-encrypted schizophrenia PRS (calculated with LDPred2) and PRS calculated from HEPRS using four different ring moduli parameters. PRS accuracy was assessed by considering individuals with >0.5 percentile scores for each calculation to have schizophrenia, and comparing with the actual phenotypes. Area under ROC (AUROC) values are equal across all methods.

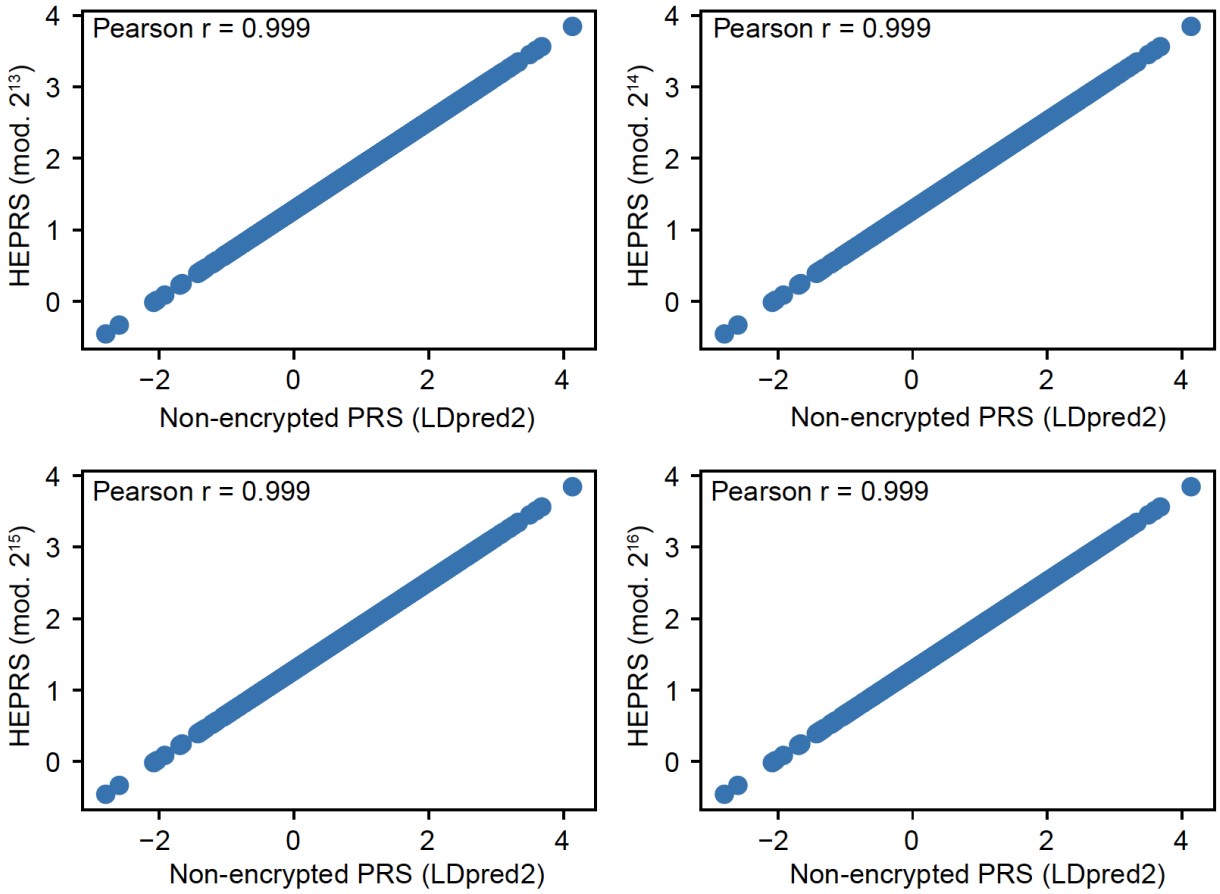

Figure 7B: Pearson correlations between the encrypted schizophrenia PRS with four different moduli parameters and the non-encrypted PRS calculated with LDPred2 each show a high correlation (r >0.999).

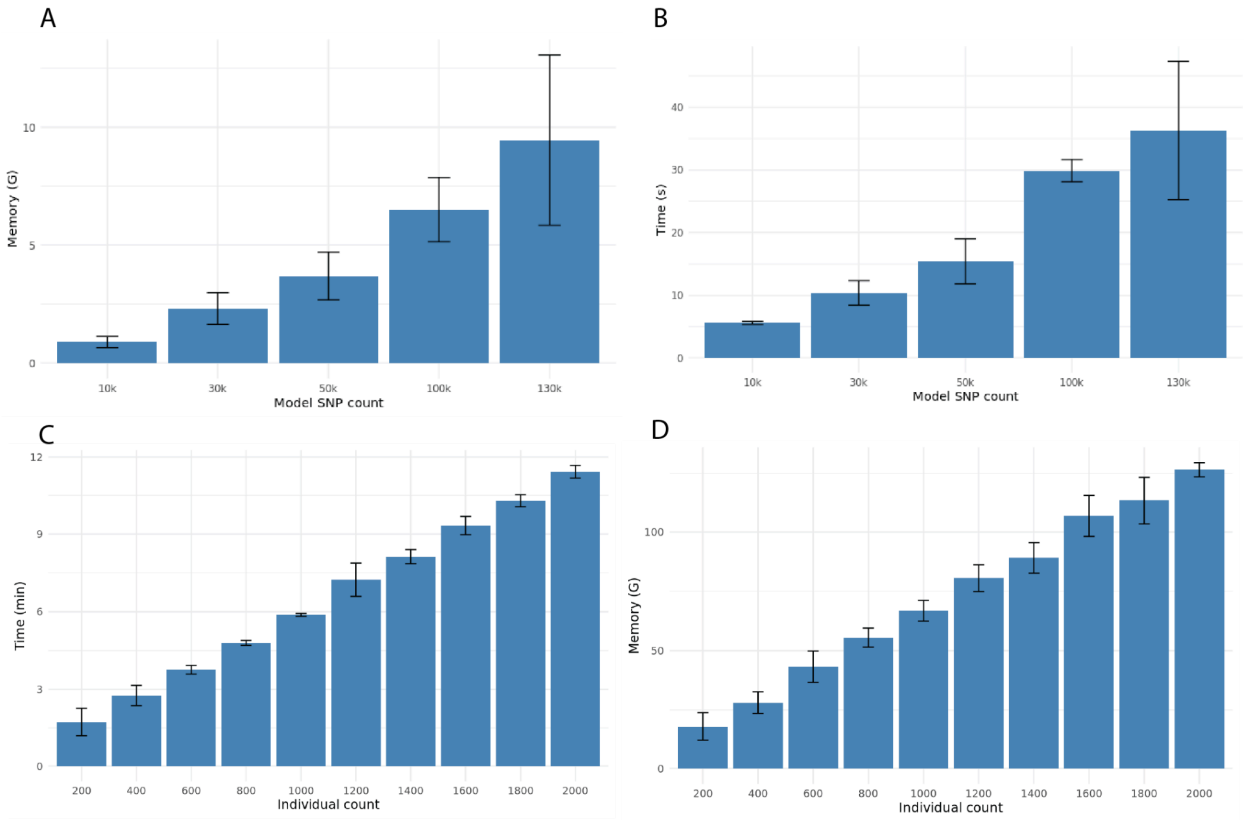

Figure 8: HE computation time and memory usage scale approximately linearly with the number of SNPs and individuals. (A) Computation time and (B) memory usage as a function of the number of SNPs, generated using synthetic datasets containing a range of SNP sizes created by Hapgen2. The runs were performed using a modulus degree of 2^{13}, a sample size of 50 individuals, and phenotype 0 (as specified in Figure 2 and Table 2). (C) Computation time and (D) memory usage as a function of the number of individuals, using the synthetic dataset with 130,000 SNPs and varying only the number of individuals for each run. A modulus degree of 2^{13} and phenotype 0 were used for these experiments. Error bars indicate the standard deviation from 10 independent trials tested on a 6234 CPU.

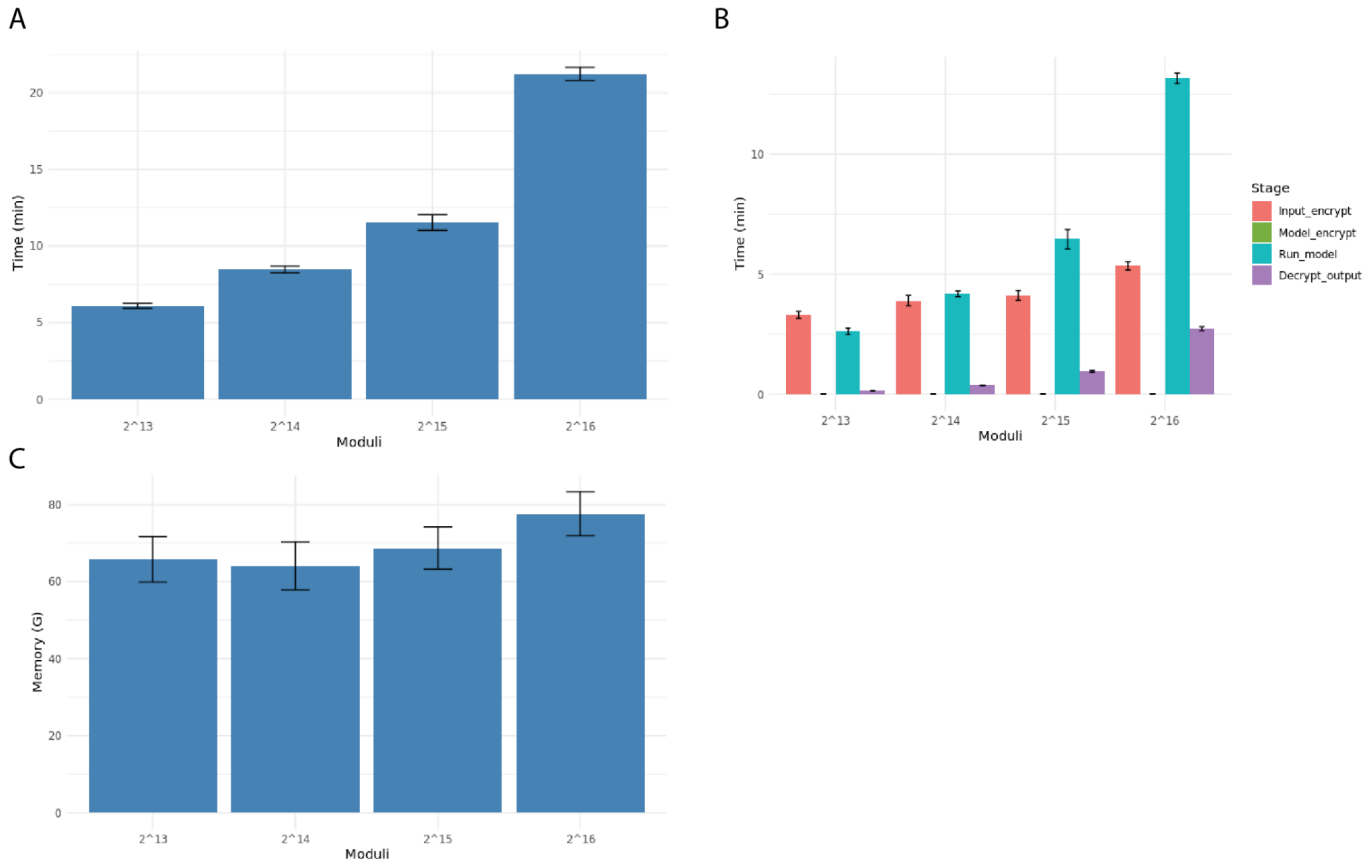

Figure 9: The effect of modulus degree on computation time and memory usage in homomorphic encryption (HE) for a schizophrenia polygenic risk score (PRS) model generated from real data. The model contains 1,146 individuals and 110,258 SNPs. The range of tested modulus degrees is from 2^{13} to 2^{16}. (A) Total computation time as a function of modulus degree. (B) Breakdown of computation time into different steps of the HE process. (C) Memory usage as a function of modulus degree. Error bars indicate the standard deviation calculated from 10 independent trials tested on a single 6234 CPU.