DyPS: Dynamic, Private and Secure GWAS

Antoine Boutet, Túlio Pascoal, Jérémie Decouchant, Paulo Esteves-Verissimo

DyPS: Dynamic, Private and Secure GWAS

Antoine Boutet
Univ Lyon, INSA Lyon, Inria, CITI
antoine.boutet@insa-lyon.fr

Túlio Pascoal
SnT, University of Luxembourg
tulio.pascoal@uni.lu

Jérémie Decouchant
Delft University of Technology
j.decouchant@tudelft.nl

Paulo Esteves-Verissimo
KAUST - RC3
paulo.verissimo@kaust.edu.sa

Abstract

Genome-Wide Association Studies (GWAS) identify the genomic variations that are statistically associated with a particular phenotype (e.g., a disease). The confidence in GWAS results increases with the number of genomes analyzed, which encourages federated computations where biocenters would periodically share the genomes they have sequenced. However, for economical and legal reasons, this collaboration will only happen if biocenters cannot learn each others' data. In addition, GWAS releases should not jeopardize the privacy of the individuals whose genomes are used. We introduce DyPS, a novel framework to conduct dynamic privacy-preserving federated GWAS. DyPS leverages a Trusted Execution Environment to secure dynamic GWAS computations. Moreover, DyPS uses a scaling mechanism to speed up the releases of GWAS results according to the evolving number of genomes used in the study, even if individuals retract their participation consent. Lastly, DyPS also tolerates up to all-but-one colluding biocenters without privacy leaks. We implemented and extensively evaluated DyPS through several scenarios involving more than 6 million simulated genomes and up to 35,000 real genomes. Our evaluation shows that DyPS updates test statistics with a reasonable additional request processing delay (11% longer) compared to an approach that would update them with minimal delay but would lead to 8% of the genomes not being protected. In addition, DyPS can result in the same amount of aggregate statistics as a static release (i.e., at the end of the study), but can produce up to 2.6 times more statistics information during earlier dynamic releases. Besides, we show that DyPS can support a larger number of genomes and SNP positions without any significant performance penalty.

Keywords Federated GWAS, Genomic privacy, Dynamic workload, Collusion resistance

1 Introduction

The decreasing cost of DNA sequencing [1] allows Genome-Wide Association Studies (GWAS) to become affordable. GWAS aim at identifying the genetic factors underlying a set of observable characteristics or traits such as diseases or particular phenotypes [2]. To do so, the genomes of a case population (i.e., a group of participants with a phenotype of interest) are sequenced to compare the statistical features of their Single Nucleotide Polymorphisms (SNPs, one nucleotide long genomic variations) to those of a control population (i.e., a group of individuals without the targeted phenotype).

Research and medicine would benefit from publicly shared GWAS statistics [3–6]. In particular, health scientists and geneticists have been using aggregate statistics (i.e., single or pairwise allele frequencies) and test statistics (e.g., χ^2-test results) for their studies. However, several privacy attacks [3, 4, 7–9] demonstrated that genomic data can only be shared with special care. Following the publication of these attacks, the National Institute of Health (NIH) preventively removed all GWAS results from public access and instantiated an approval process one has to follow to consult them [10].

Few works have studied how to determine when GWAS results can be safely released [11, 12]. However, these approaches require gaining access to the full set of genomes, which is not realistic in modern federated biomedical ecosystems [13–15], where each biocenter keeps its data and supports the cost of its own sequencing process, which limits full data sharing for economical and legal reasons. Therefore, there is now a need for federated GWAS solutions that would encourage collaboration of players with different economical interests to increase the accuracy of GWAS, distribute their costs and reduce their computational delays.

In this context, several works have presented methods that enable the distributed computation of aggregate statistics, relying either on homomorphic encryption (HE) [16–19], secure multi-party computations (MPCs) [20–22], or secret-sharing [23]. However, these works do not consider adversarial environment where GWAS results might be publicly released and potentially attacked by an adversary who might control one or several of the biocenters.

In addition, to speed up the release of the first results, GWAS could clearly benefit from a dynamic process of updating the results gradually as the number of genomes evolves. Indeed, GWAS can involve up to 300,000 SNPs and thousands of subjects [16], in which case it may be necessary to wait for a long time before accessing the final results.

Finally, there is a pressure to enable individuals to control how their data are used [24, 25]. To comply with current data-privacy regulations constraints such as the US HIPAA and EU GDPR, data subjects shall have the right to withdraw

their consent to participate in a GWAS at any time. Enforcing privacy in this context is challenging since a potential adversary having access to several GWAS result releases could leverage the evolution of the results to infer data.

Dynamic GWAS is therefore facing several challenges that are difficult to address simultaneously: i) results should never allow any infringement on the privacy of their users; ii) users should be able to retract their consent at any time; iii) the data of each biocenter have to be secured (i.e., used to produce results but not revealed even when facing collusion between parties); and iv) global results should be publicly accessible and updated as soon as possible. Our work is the first to simultaneously address all these challenges.

Contributions. We present Dynamic, Private and Secure GWAS (DyPS), a federated system where biocenters collaborate by dynamically and safely contributing data to a GWAS without losing control over the genomes they have sequenced. DyPS leverages a Trusted Execution Environment (TEE), i.e., Intel SGX, to securely process encrypted genomes and compute GWAS statistics. In addition, DyPS ensures privacy-preserving GWAS while accepting participation consent withdrawal. To do so, DyPS implements efficient algorithms that determine how to safely release and update GWAS statistics without noise addition, which guarantees no data utility loss. Moreover, DyPS tolerates up to all but one colluding biocenters without privacy leaks.

We evaluated DyPS with a synthetic GWAS workload studying up to 300,000 SNPs and more than 6 million simulated genomes. We also evaluated DyPS using two different genomic datasets consisting of 2,000 and 35,000 real genomes, respectively. Our results show that DyPS dynamically updates test statistics with a reasonable additional request processing delay (11% longer) compared to a naïve approach that would update them with minimal delay but would lead to 8% of the genomes not being protected. Moreover, DyPS eventually released the same amount of aggregate statistics than a static release method, which would release statistics after all genome requests have been processed, and punctually released up to 2.6 times more statistics (44 instead of 17 for 350 considered SNPs). Finally, DyPS tolerates colluding biocenters. In the worst case scenario (only one biocenter is not colluding), DyPS updates the GWAS results with only 6.6% and 21.4% fewer genomes, for addition and removal requests, respectively, when compared to a scenario without collusion. In addition, DyPS can support a larger number of genomes and SNP positions without any significant performance penalty.

This paper is organized as follows. Section 2 provides a comprehensive background on GWAS, the known attacks that take their results as input, and their countermeasures. Section 3 presents the system and threat models we consider, along with our objectives. Section 4 provides an overview of DyPS' architecture and workflow, and presents the methods we designed for the release and update of GWAS results in a federated and adversarial environment. Section 5 evaluates DyPS' performance. Section 6 surveys the related work. Section 7 concludes.

2 Background

2.1 Genotype encoding

Since any two genotypes (i.e., a complete set of genes) are mostly identical, one can represent a genotype as its differences with a reference genome. GWAS consider a particular subset of these differences, Short Nucleotide Polymorphisms (SNPs), which are genomic variations that involve a single nucleotide. For example, at a specific genome location, most individuals may have a given nucleotide, e.g., a C, while others may have an alternative nucleotide, e.g., an A. The two possible nucleotide variations are called alleles (or variants). Table 1 provides an example of N genotypes $\{g_1, \ldots, g_N\}$ that are described over L variants $\{\text{SNP}_1, \cdots, \text{SNP}_L\}$. A "1" in this record represents the fact that the associated genome contains the least frequent allele (minor allele) of the corresponding SNP.

Table 1. Genomes encoding.

	SNP_1	SNP_2	SNP_3	…	SNP_L
Genome g_1	0	1	0		1
g_2	1	1	0		1
⋮				⋱	
g_N	0	1	1		0

2.2 Aggregate and test statistics

GWAS may produce: (i) *aggregate* statistics, such as minor allele frequencies (MAF), single and pairwise allele frequencies; and (ii) *test* statistics, such as linkage disequilibrium (LD), χ^2 and *p-values*. Aggregate statistics report the counts of each allele or pair of alleles. Going further, test statistics aim at quantifying the confidence one can have when interpreting the data, e.g., when correlating a SNP with the observed phenotype.

Table 2. A singlewise SNP contingency table.

		Phenotype		
		Population		
		Case	Control	Total
SNP_l	0 (major)	N_0^{case}	$N_0^{control}$	N_0
	1 (minor)	N_1^{case}	$N_1^{control}$	N_1
	Total	N^{case}	$N^{control}$	

Aggregate statistics. A prerequisite for the production of all GWAS statistics is the computation of allele contingency tables, which contain the allele counts for each SNP in each population (single allele frequencies). Table 2 illustrates the contingency table associated to SNP_l. N_i^{pop} is the count of allele $i \in \{0, 1\}$ in population $pop \in \{\text{case}, \text{control}\}$. N^{case} and

$N^{control}$ are, respectively, the size of the case and the control population. N_0 and N_1 are, respectively, the overall counts of the major and minor alleles. The MAF is the frequency of the least common allele of a SNP in a population, e.g., N_1^{case}/N^{case} for the case population. The pairwise allele frequencies of two variants SNP_{l_1} and SNP_{l_2} ($C_{--}^{l1,l2}$) report the number of occurrences of the four possible combinations of alleles $\{00, 01, 10, 11\}$ in a population $pop \in \{\text{case}, \text{control}\}$ (Table 3).

Table 3. A pairwise contingency table for two variants, SNP_{l_1} and SNP_{l_2}, where $l_1, l_2 \in \{1, \cdots, L\}$.

		SNP_{l_2}		
		0	1	Total
SNP_{l_1}	0	$C_{00}^{l1,l2}$	$C_{01}^{l1,l2}$	$C_{0-}^{l1,l2}$
	1	$C_{10}^{l1,l2}$	$C_{11}^{l1l,2}$	$C_{1-}^{l1,l2}$
	Total	$C_{-0}^{l1,l2}$	$C_{-1}^{l1,l2}$	$2N^{pop}$

Test statistics. Several metrics have been defined to identify whether the co-occurrences of two alleles can be considered random. These metrics can be computed from the pairwise allele frequencies of the two variants. For example, the value of the linkage disequilibrium metric D can be computed as follows:

$$D = \frac{C_{00}^{l1,l2}}{2N^{pop}} - \left(\frac{C_{0-}^{l1,l2}}{C_{0-}^{l1,l2} + C_{1-}^{l1,l2}} * \frac{C_{-0}^{l1,l2}}{C_{-0}^{l1,l2} + C_{-1}^{l1,l2}} \right)$$

Differently, the χ^2 hypothesis test determines whether or not the null hypothesis, which states that the allele frequencies in the case and control populations follow a similar distribution, can be rejected. The χ^2 statistic of a single SNP is defined as:

$$\chi^2 = \sum_{i \in \{0,1\}} \frac{(N_i^{case} - N_i^{control})^2}{N_i^{control}}$$

From the value of the χ^2 statistic, one can then compute the *p-value* of each SNP, which is the probability of observing its contingency table should the null hypothesis be correct. A *p-value* smaller than a given threshold (e.g., 10^{-7}) typically indicates that the variant might play a significant role in the observed phenotype.

2.3 Privacy attacks

An adversary may try to leverage a GWAS's metadata (i.e., lists of SNPs and pseudonymized genomes) and test and/or aggregate statistics to breach the genomes' owners privacy. Figure 1 illustrates the typical information that an adversary can observe: (i) the list of the L SNPs; (ii) the N genome pseudonyms used in the GWAS, and (ii) the GWAS results which may include one or several of the statistics introduced in Section 2.2. To be noted, that if the adversary is a biocenter contributing to the GWAS computation, this adversary knows a subset of the SNPs and genome pseudonyms as well as a subset of the content of the table. This adversary knowledge increases in case of collusion between several biocenters. In the following, we detail the two categories of attacks the adversary might try to execute, namely recovery and membership attacks.

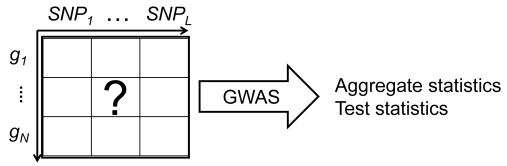


Figure 1. Observable data for privacy attacks on GWAS.

Recovery attack. A recovery attack aims at reconstructing the allele sequences of the individuals who participated in the study (i.e., the content of Table 1). Inferred genotypes might allow the unwanted reidentification of the subjects who participated in the study [26]. In addition, sequencing individuals represents a financial effort, and private biocenters would refrain from participating in a study if there was any risk of seeing their data being inferred by competitors.

Membership attack. In a membership attack an adversary aims at determining whether a genotype g_i it already knows is included in the case population [7]. Upon success, the adversary infers that the victim has the studied phenotype (e.g., a disease). Several membership attacks have been implemented in the literature, for example on genotype frequencies [8], and SNPs correlation [3]. More recently, Cai et al. [4] showed that it is possible to identify individuals in the case group of a GWAS knowing their genomic variations at only 25 randomly selected SNP positions.

2.4 Conditions for safe releases

Different conditions have to be respected to ensure safe releases of GWAS results (i.e., to avoid inference attacks from statistics). These conditions differ according to the statistics produced by the GWAS and the targeted attack (Table 4). These conditions are always enforced in DyPS, which is a challenge given the fact that DyPS dynamically releases GWAS results, contrary to previous works, which in turn cannot be directly applied.

Test statistics. Zhou et al. have shown that the recovery and membership attacks based on test statistics are NP-complete problems [12]. Building on this result, they argued that GWAS results can be safely released when the possible solution space (i.e., the number of possible combinations of genotypes) is significantly larger than the original GWAS result space (e.g., the pairwise frequencies space). Practically, it comes down to having limits on the number of genome manipulated (Equations (4) and (5)).

Aggregate statistics. The likelihood ratio (*LR*) [11], on one side, and the T_r [3] and Λ [12] metrics, on the other side, can

Table 4. Release conditions for GWAS aggregate or test statistics computed over L SNPs and N individuals.

Observable statistics	Attack	Attack unfeasibility conditions
Aggregate	Membership	*Single allele freq.:* LR metric is sufficiently low, $N > 100$ *and* MAF > 0.05 (1) *Pairwise allele freq.:* Λ or T_r metric is sufficiently low. (2)
	Recovery	$2(N-1)/\log(N+1) > L$ (3)
Test	Membership	$2N/(\log(N+1)-1) > L$ (4)
	Recovery	$2(N-1)/(\log(N+1)-1) > L$ (5)

be used to bound the identification power achievable over, respectively, single and pairwise aggregate statistics. Building over the LR, SecureGenome [11] calculates the sensitivity and the specificity of the LR-test to decide over which SNP positions allele frequencies can be safely released. We use a similar approach on singlewise aggregate statistics, and extend it to pairwise aggregate statistics (see Section 4.4). Similarly to test statistics, a bound on the number of manipulated genomes avoids recovery attack from aggregate statistics (Equation (3)).

3 Models

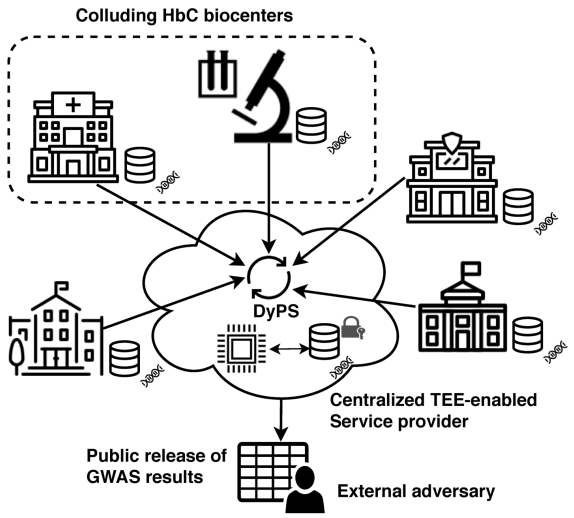


Figure 2. DyPS' system and threat model.

System model. Figure 2 illustrates our system and threat models. We consider a federated system of B biocenters $\{b_1, \cdots, b_B\}$, which obtain and store locally the allele sequences of individuals. Each biocenter may sequence the genomes of case and/or control individuals, potentially at different speed rates due to various model of sequencing machines. We assume that biocenters can contribute a genome to a GWAS, and that individuals retract their participation consent, at most once. The biocenters are connected through an asynchronous communication network. Their common goal is to perform a GWAS over a set of L pre-selected SNPs. We denote by $LtoN(L)$ the minimum number of genomes that need to be used to release statistics computed over L SNPs so that Equations (3), (4) and (5) in Table 4 are enforced. Reciprocally, we denote by $NtoL(N)$ the maximum number L of SNPs that can be safely released according to the number of participating genomes N.

We assume the availability of a server equipped with a TEE dedicated to the GWAS. Consequently, it might identify the pseudonyms of the genomes that are used for a statistics release. We assume accordingly that the parties have access to this information since the pseudonyms of the used genomes are made public (cf. Section 2.3). This TEE is responsible for executing the actual GWAS computation, and ensuring that the result is safe before releasing it in open access.

Threat model. We assume a probabilistic polynomial time adversary that has access to a good reference population with an allele distribution identical to that of the case population. To launch membership attacks, the adversary also has access to a victim's DNA profile (i.e., its genetic code).

We consider an honest-but-curious adversary controlling biocenters and monitoring the GWAS releases: biocenters follow the protocol and do not forge genomes. In addition, we consider collusion between biocenters that aim at inferring information from the non-colluding biocenters. We assume that up to $f = (B-1)$ biocenters can collude to launch either membership or recovery attacks on the released GWAS results. We leave the investigation of additional adversarial behaviors for future work. Indeed, enforcing genomic data genuineness is an open challenge as digital genomes can be forged [27, 28] or synthesized [29], which would make data poisoning attacks undetectable. Even though all data is stored encrypted on the TEE server and only manipulated by the enclave, DyPS does not cope with possible Intel SGX side-channel attacks [30].

4 DyPS: Dynamic, Private and Secure GWAS

DyPS adopts a federated architecture that allows each biocenter to safely share genomes through a server computing GWAS, while keeping the control of their own genomes (i.e. without revealing their data to other biocenters and by ensuring no leakage from GWAS results). This architecture is illustrated in Figure 3. To ensure a secured computation potentially performed by untrusted machines, DyPS relies on a TEE which leverages custom microprocessor zones, to enforce isolation, confidentiality and integrity of both the data and operations. Periodically, the enclave collects the requests from the various biocenters and decides which requests are to be executed to safely and dynamically update

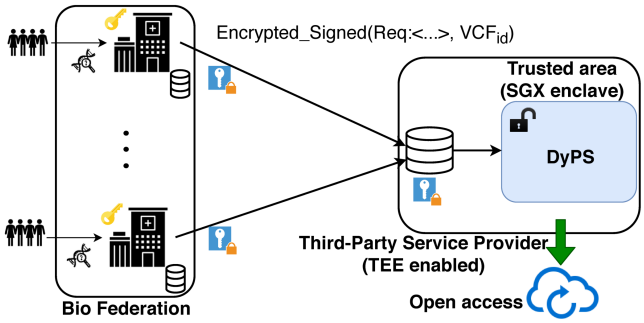


Figure 3. DyPS' federated architecture.

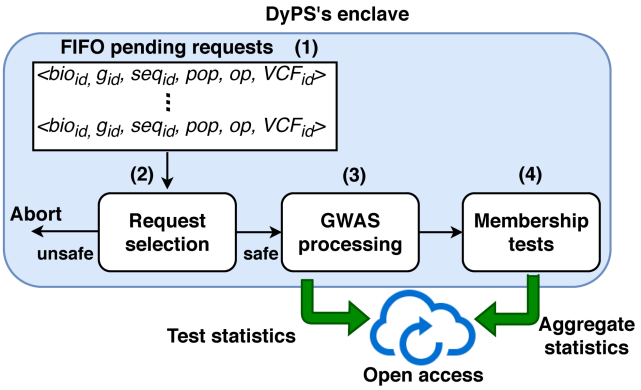


Figure 4. DyPS' workflow diagram.

the GWAS results. In the following, we describe how the TEE is exploited (Section 4.1), the workflow of DyPS (Section 4.2), how batches of requests are selected to produce safe test statistics (Section 4.3), the production of aggregate statistics from the selected requests (Section 4.4), and how DyPS can dynamically increase the number of SNPs over which statistics are computed (Section 4.5).

4.1 TEE-based architecture

DyPS uses Intel Software Guard Extensions (SGX) [31], which defines the concept of enclave as an isolated unit of data and code execution that cannot be accessed even by privileged code (e.g., the operating system or hypervisor). Enclaves can be attested to prove that the code running in the enclave is the one intended, and that it is running on a genuine Intel SGX platform. Once attested, enclaves can be provisioned with secret data by using authenticated secure channels. Moreover, enclaves can persist secret data outside the trusted zone by using a sealing mechanism.

Once DyPS' enclave has been initialized, each biocenter executes a remote attestation procedure to authenticate it and establish a secret symmetric key. The biocenters sign their data with their private key, encrypt it with the shared symmetric key and send it over the network to the enclave's host. Upon reception of the encrypted data by the untrusted host, the enclave loads it into its protected memory space and decrypts it.

As time goes by, the biocenters are expected to sequence genomes, and can receive participation consent withdrawals. For each genome addition or removal, the biocenters send a request to the enclave. This request $\langle bio_{id}, g_{id}, seq_{id}, pop, op, \mathtt{VCF}_{id}\rangle$ contains the biocenter ID (bio_{id}), the donor's pseudonym (g_{id}), the operation sequence number of the biocenter (seq_{id}), if the donor belongs to the control or case population (pop), whether the genome should be added or removed ($op \in \{Add, Rmv\}$), and the corresponding genotype data following the Variant Cell Format file format ($\mathtt{VCF}_{id}$) in case of genome addition.

4.2 Workflow diagram

GWAS can produce both test statistics and aggregate statistics. DyPS follows the workflow depicted in Figure 4 to ensure safe releases in both cases. This workflow is executed in the enclave and contains multiple modules: (1) the pending requests queues, (2) the request selection to produce safe test statistics, (3) the GWAS processing, and (4) the test to produce safe aggregate statistics.

(1) FIFO pending requests. DyPS maintains FIFO queues of genome additions or removals for each biocenter. DyPS tries to execute the received requests according to their initial ordering by the biocenter through the use of their sequence number. However, DyPS might not always treat requests across the FIFO queues (e.g., because a genome A cannot be immediately removed, while a genome B can be added). A particular case occurs when a request to remove a genome that has not yet been added to the GWAS is received, in which case, both requests can immediately be executed by removing them from the FIFO queues.

(2) Requests selection. During this phase, DyPS aims at identifying a subset of genome operations that can be safely executed to update the GWAS results. To avoid exhaustive search of safe operations, DyPS assembles batches of requests where each set of non-colluding biocenters ($B-f$) contributes more genome additions than removals, and sufficiently enough requests overall. The genome additions and removals to be executed are selected according to their FIFO order. We detail the algorithm we use in Section 4.3 and in Appendix A, and prove that it prevents all privacy leaks on test statistics. If no set of requests can be processed to release statistics, the process aborts. Since DyPS is periodically executed, requests are eventually processed.

(3) GWAS processing. After collecting a batch of requests that verifies Equations (3), (4) and (5) in Table 4, DyPS computes the GWAS results over the overall remaining genomes in the SGX enclave. If the GWAS only aims at releasing test statistic, at this point DyPS can safely release

or update the publicly accessible results (i.e., skipping step (4)).

(4) Membership tests. If DyPS aims at computing aggregate statistics, this additional step verifies that membership attacks cannot be executed on aggregate statistics (Equations (1) and (2) Table 4). To do so, DyPS relies on the *LR* and the Λ metrics in a SNP selection algorithm, which exhaustively verifies that releases never introduce privacy leaks (cf. Section 4.4 and Appendix B).

4.3 Request selection to address test statistics

We now detail how biocenters can add or remove genomes from the results of a GWAS assuming (for now) that the number of studied SNPs remains constant. Both operations, if not handled carefully, can create privacy issues when the released statistics are updated. For example, updating statistics by adding, or removing, a single genome might directly leak this genome to an adversary that would observe publicly released statistics.

An exhaustive search (i.e., a brute force approach) checking if any candidate set of selected requests combined with the sets of requests used in previous releases verify Equations (3), (4) and (5) in Table 4 is not practical and would require exponential time. To avoid this issue, DyPS waits to have received sufficiently enough requests from the biocenters to verify equations listed in Table 4. More specifically, DyPS uses equation (3) as it implies equations (4) and (5).

DyPS assembles batches of genome operations according to the FIFO ordering of requests, and so that a batch contains more additions than removals for every subset of $(B - f)$ biocenters, and an overall number of genome operations either equal to 0 or larger than $LtoN(L)$ for every subset of $(B-f)$ non-colluding biocenters. We provide the pseudocode of DyPS' requests selection algorithm, and prove by induction that the adversary is never able to isolate test statistics where less than $LtoN(L)$ genomes participate in Appendix A.

From a high-level perspective, this algorithm works as follows. First, all pending addition requests of biocenters are selected, and for each of them, at most an equal number of pending removal requests. Then, the biocenters with the smallest number of selected requests are eliminated, until the $B-f$ biocenters with the least numbers of operations collectively possess enough requests (i.e., more than $LtoN(L)$) or until each biocenter has enough requests by itself. All requests from the biocenters that have not been discarded participate in the next release. This algorithm has a low complexity, since the previous statistic releases are not considered, while a brute force algorithm would have a exponential complexity with the number of previous releases (as shown in Section 5.2). The test statistics can then be dynamically updated using the selected requests.

4.4 Membership tests to address aggregate statistics

Aggregate statistics are computed over a SNP. Consequently, DyPS only includes in the GWAS results the SNPs which depicted safe aggregate statistics. Similarly to provide safe test statistics, preventing recovery attacks from aggregate statistics relies on enforcing a minimum batch size of genome (i.e., Equation (3) in Table 4). However, to provide aggregate statistics also preventing membership attacks, the current batch of requests and its combinations with previous releases must verify conditions (1) or (2) of Table 4. To ensure these conditions, DyPS relies on metrics that bound the identification power achievable over a set of requests and GWAS result (i.e., *LR* for singlewise, and/or Λ for pairwise frequencies) to identify over which SNPs to update the GWAS results. More specifically, given a set of genomes, a set of SNPs, and a set of control genomes (the adversary knowledge), DyPS determines which SNP positions can have their allele frequencies safely released by firstly removing very rare allele frequencies (MAF $\leq$ 0.05), and SNPs in high linkage disequilibrium (*p-value* below 10^{-5}) among the participating SNPs. After this step, DyPS computes and evaluates the detection power achieved by the singlewise *LR* or pairwise Λ in order to decide over which SNPs aggregate statistics can be safely released. It should be noted that DyPS reproduces SecureGenome [11] on singlewise allele frequencies inside a TEE, and extends it by also considering pairwise allele frequencies using the Λ metric (cf. Appendix B).

Identifying a set of SNPs to update in a given batch of genome operations however does not guarantee that the privacy of each genome will never be breached. Indeed, any release of aggregate statistics can be combined with past releases, and the genomes that a subset of up to f colluding biocenters contributed can be removed from the resulting aggregate statistics.

To verify whether a SNP's single or pairwise allele frequencies can be updated with a batch of genome operations, DyPS executes an exhaustive verification. More precisely, for a given SNP, this exhaustive verification (i) gathers all releases where statistics over the selected SNP have been released, and (ii) verifies that any combination of these releases have a low enough *LR* or Λ score for the given SNP for any combination of up to f adversary biocenters. However, the space complexity of this check remains constant. The complexity of this verification scheme is $O(L' \cdot 2^R \cdot \binom{B}{f})$, where L' is the number of selected SNPs in the current candidate batch, and R is the current number of releases. In practice, one could simply tune DyPS to limit the maximum number of releases it wishes to avoid spending too much time performing verifications. We further discuss the exhaustive verification procedure in Appendix B.

Figure 5 illustrates some scenarios DyPS might face with aggregate statistics. In this example, DyPS determined a batch of N_1 genomes over which statistics might be released

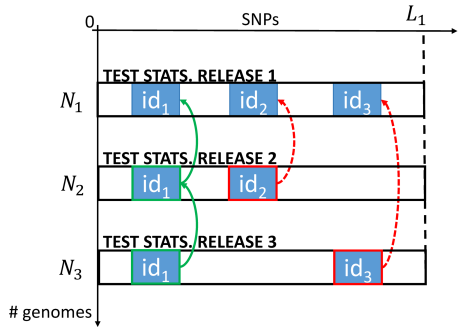


Figure 5. Successive releases of test and aggregate statistics as new genome addition or removal requests are executed.

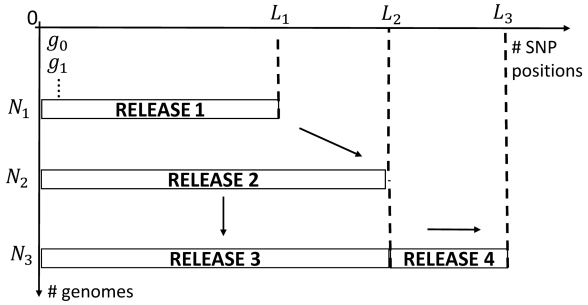


Figure 6. SNPs set dynamic scaling.

according to the method we defined for test statistics, over L_1 SNP positions. Using the SNP selection algorithm, DyPS determines that aggregate statistics over a subset (id_1, id_2, id_3) of those $L' = 3$ SNPs can be released (release 1). In release 2, following the same algorithm, DyPS determines that aggregate statistics can be released over $L' = 2$ SNPs (id_1 and id_2). DyPS then verifies whether each combination of previous releases with the current one still allows aggregate statistics to be released, i.e., the combination of releases 1 and 2. This verification passes for SNP id_1 (represented with plain arrows), which means that the statistic can be updated, while it does not for SNP id_2 (represented with dashed arrows), which cannot be updated during this release. Over release 3, DyPS determines that L' SNPs (id_1 and id_3) can be released over the selected genomes. However, the verification process identifies that the aggregate statistics cannot be released for SNP id_3 (because of the combination of releases 1 and 3, dashed arrow), while all verifications pass for SNP id_1 that can be updated (all combinations of releases are not shown for simplicity).

4.5 Scaling the GWAS over number of SNPs

So far, we have assumed that statistics are computed over a static set of SNPs. However, as more genomes become available, DyPS can dynamically increase the number of SNPs over which statistics are computed, as illustrated in Figure 6.

The initial statistics release (i.e., release 1 in Figure 6) happens when the enclave can assemble a batch of N_1 genome addition requests such that $L_1 = LtoN(N_1) \geq 1$. DyPS then automatically decides the subsequent releases, based on the conditions of Table 4, as follows. Let us assume that the i-th release of statistics decided by DyPS covers L_i SNPs, and let $N_i = NtoL(L_i)$. The number of genomes N_{i+1} and the number of SNPs over which to release statistics L_{i+1} are determined as follows. First, N_{i+1} must verify $N_{i+1} - N_i > NtoL(L_i)$, which states that the statistics over the first L_i SNPs need to be sufficiently updated to preserve the privacy of the newly considered individuals over these SNPs. The value of L_{i+1} is then computed using $L_{i+1} = LtoN(N_{i+1} - N_i)$. We call this process a diagonal expansion (release 1 to 2 in Figure 6). The actual composition of the requests may contain additional genome removals. In that case, DyPS uses the methods we have defined previously for test and aggregate statistics over the SNPs considered both by release i and $i+1$ to prevent privacy leaks.

DyPS can also handle two additional special cases. The first one happens when $L_{i+1} = L_i$, which can happen when release L_i updated the full set of studied SNPs (vertical expansion, release 2 to 3 in Figure 6). The second case happens when the value of L is increased by the system administrator. In that situation, the number of SNPs over which statistics are released might be increased immediately if the number of genomes added allows it (horizontal expansion, release 3 to 4).

5 Performance evaluation

5.1 Experimental setup

We used both Windows 10 Enterprise and an Ubuntu 18.04 LTS in a 64-bit machine, equipped with 16 GB of RAM and an Intel i7-8650U @ 2.11.GHz, which supports Intel SGX. We evaluate the performance of DyPS under several scenarios using simulated and real genomes. We run DyPS' code both in Java using Java JDK 12.0.1 and Eclipse IDE (4.11.0), and inside an Intel SGX enclave using Graphene [32] to implement DyPS in C++, so that it can run inside the SGX enclave. We use AES 256 to encrypt messages, and ECDSA for signatures. When it executes the remote attestation procedure, a biocenter agrees on a key with the enclave, which it uses to encrypt and sign the data it sends to the enclave, while the enclave can verify it upon reception.

During the experiments, we use rounds where biocenters generate requests that are sent to the enclave, and the enclave tries to generate a GWAS statistics release. In real settings, rounds would typically have a one day duration. The biocenters use a Poisson distribution to generate genome addition or removal requests. We set the parameters of these Poisson distributions so that biocenters generate more genome additions than removals, as we expect it to reflect reality. For

the experiments, based on simulated or on real genomes, we assumed a default $\lambda = 8$ for additions, and $\lambda = 6$ for removals as default. For larger GWAS settings, we have proportionally increased λ. We have adopted such values based on the increasing rate of genome sequencing, and the growing concern about genome privacy risks among society nowadays. For example, Dankar et. al [24] have recently evaluated and claimed the need for the creation of dynamic information consent models capable of autonomously enabling individuals to opt out of participating in genomic studies at any time.

We compare the performance of DyPS' request selection heuristic to a brute force (*BF*) and a naïve algorithm. The *BF* approach aims at adding or removing genomes by assembling batches of genome operations, and checking whether they are safe by combining them with all previous combinations of data releases. DyPS scales better than the BF algorithm with the number of data releases by avoiding this brute force verification for test statistics. The naïve approach waits for biocenters to collectively have $LtoN(L)$ genomes (additions or removals) when performing a release. This method can be seen as the current state-of-the-art, and does not allow genomes removals. We show that this method executed with genome removals leads to privacy risks.

Regarding aggregate statistics, since DyPS is the first dynamic GWAS protocol, we can only compare it to a static GWAS algorithm, which would wait for all requests to have been collected before releasing statistics. To do so, we compare DyPS with a static approach that would rely on the *LR* metric [11] to release singlewise allele frequencies at the end of the experiment. For this experiment, we used the default *LR* parameter values defined in [11]: a MAF cut-off of 0.05, a LD between SNPs cut-off of 10^{-5}, a false positive rate of 0.1, and a true positive rate of 0.9.

We stress DyPS by simulating the generation of up to 6 million genomes studied over up to 300K SNPs to evaluate its performance with test statistics. In addition, we use two real genome datasets to evaluate DyPS' performance on both test and aggregate statistics: the idash2017 dataset [33], which consists of 2,000 genomes, and the phs001039.v1.p1 dataset from dbGAP [6], which consists of more than 35,000 genomes.

5.2 Bandwidth, CPU and memory consumption

We use 64-bits integers to encode the ID fields in a request, except for the *pop* and *op* fields that only require one bit. Overall the size of a request is 258 bits, which represents approximately 48 Bytes after encryption. A genome is encoded using 2 bits per SNP, which would represent only 75 KB for a GWAS studying 300,000 SNPs. Given those numbers, bandwidth is not a bottleneck for DyPS since communications only occur when biocenters asynchronously send requests to the enclave.

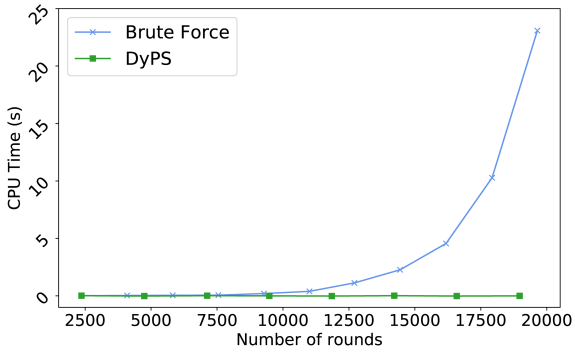


Figure 7. Running time of the brute force and DyPS request selection approach for test statistics over 5,000 SNPs ($LtoN(L) = 38,040$) ($B = 4, f = 0$).

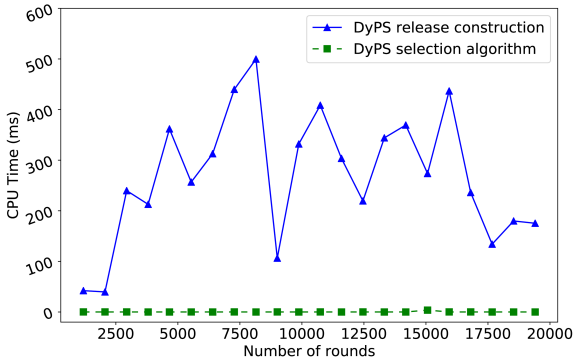


Figure 8. Running time of DyPS request selection approach for a test statistics over 5,000 SNPs ($LtoN(L) = 38,040$) inside the SGX enclave ($B = 4, f = 0$).

To measure DyPS' CPU and memory consumption, we first consider a GWAS scenario that involves 4 non-colluding biocenters ($B = 4, f = 0$). Figure 7 shows the CPU running time of the brute force (*BF*) and of DyPS' request selection algorithms during a synthetic GWAS that involves 5,000 SNPs and 20,000 rounds. DyPS' SNP selection algorithm has a constant complexity and is very fast (less than 350 ms), while the *BF* selection algorithm, with exponential complexity, requires more than 20 seconds after 20,000 rounds.

Figure 8 shows DyPS' performance when deployed in an enclave. Every point represents a release that took place during the experiment. As can be noted, DyPS's selection algorithm for test statistics has a constant running time (and below 1 ms). In addition, the release construction running time varies according to the number of requests. The longest release construction running time was below 500 ms in a release with 38,059 genome operations, and the average during the experiment was 269 ms. With similar settings, we also simulated a GWAS studying 300,000 SNP positions and four biocenters and executed DyPS inside an SGX enclave.

Although DyPS was able to add 6,078,551 genomes and to remove 899,278 genomes, we observed similar behavior for the CPU running time.

We also monitored DyPS' memory consumption in the SGX enclave assuming different system settings (i.e., varying number of B and f). We did not observe significant change in memory consumption, which stays around 2 MB per round when the numbers of participating and colluding biocenters evolve. This is expected, since genomes are stored encrypted outside of the enclave and at a given time, only a limited number of genomes are loaded into the enclave's memory to be processed.

5.3 Naïve dynamic release vs. DyPS

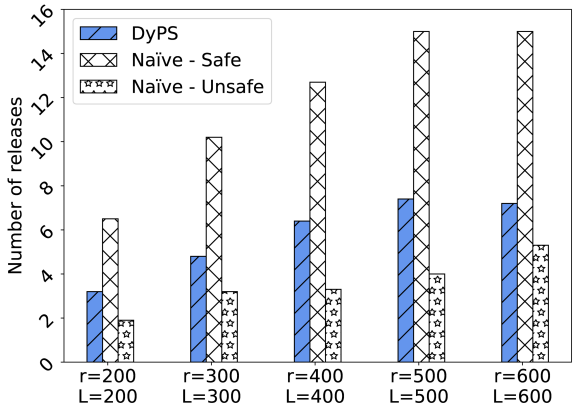
We then compare the number of releases that can be performed by the naïve approach, which updates GWAS results as soon as more than $LtoN(L)$ genomes are collected without making sure that the combination of these releases are also safe, to the number of releases that DyPS performs. Figure 9 reports the number of releases for both approaches, where the label of a bar plot reports the number of rounds for which the experiment ran (i.e., the value of r), and the number of SNPs per GWAS (i.e., L).

Figure 9a shows the corresponding number of releases done by each approach, and for the naïve approach shows the number of releases for which a genome was at risk. Up to 4.98% of the releases contained at least one genome that was at risk. Figure 9b shows the number of genomes that were added, or removed by each method during the experiments. It also shows how many genomes were at risk with the naïve release approach. Overall, the naïve approach was only able to consider at most 11% more genomes than DyPS, and exposed up to 8% of the genomes to privacy leaks (i.e., recovery attacks). Even though DyPS updates the GWAS results less frequently, which is to be expected, it is overall of low consequence on the number of considered genomes. In addition, DyPS enforces that no genome is at risk.

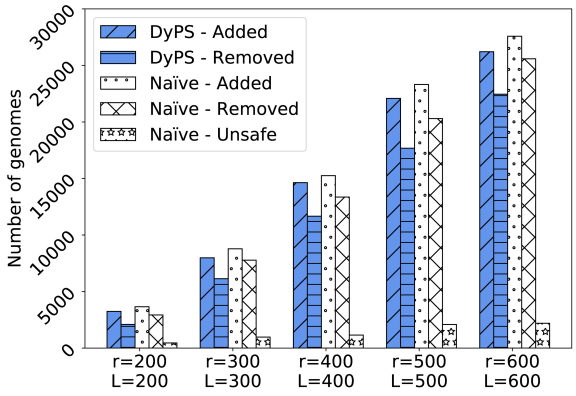
5.4 Impact of dynamic SNP set scaling

To measure how DyPS' dynamic approach reduces the treatment delay of requests, we measured the effect of the dynamic SNP set scaling mechanism. In particular, we simulated a situation where the number of rounds is limited, and L (the number of SNPs) is high, such that DyPS without dynamic scaling could create only a limited number of releases whereas DyPS was able to create more and earlier releases.

Figure 10 provides the round delays for both approaches per type of genome operation (addition or removal). DyPS without the dynamic release mechanism was able to perform a *single* release (additions of 21,602 genomes), represented by the square at the 908th round (with an average round delay above 430 rounds), during the whole experiment, but could not remove any released genomes. On the other hand, DyPS with dynamic scaling was able to execute 11 secure releases,



(a) Average number of releases.



(b) Average number of added, removed and unsafe genomes.

Figure 9. Comparison between the naïve release approach and DyPS under different scenarios (r rounds, and L SNPs) for ($B = 4, f = 0$).

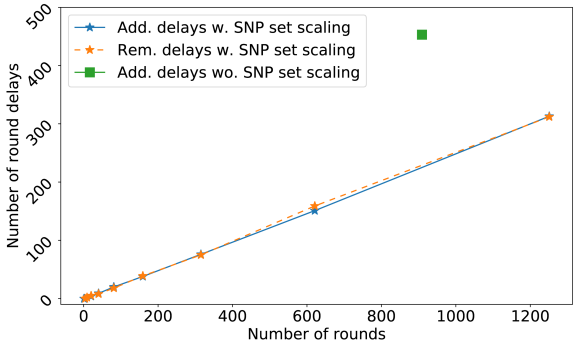
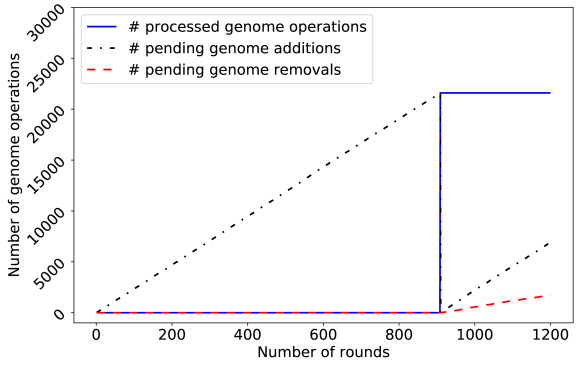


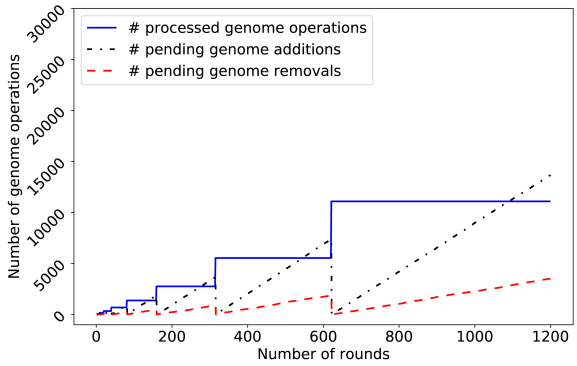
Figure 10. DyPS with or without dynamic SNP set scaling–round delays for a GWAS consisting of 3,000 SNP positions ($LtoN(L) = 21,600$) ($B = 4, f = 0$).

varying among diagonal, vertical and horizontal releases. One can also observe that DyPS treated genome additions and removals with very similar delays (the star markers),

and that the dynamic mechanism has an order of magnitude lower delay. One can also notice that the time to release of requests increases as the number of SNP positions L increases over time, even when using dynamic SNP set scaling since more genomes operations are required per batch of requests to ensure privacy.



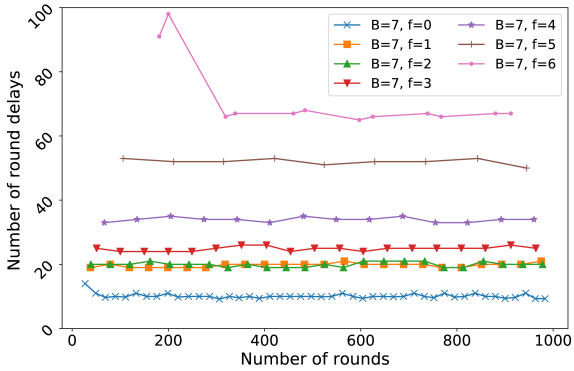
(a) DyPS without dynamic scaling of the SNPs set.



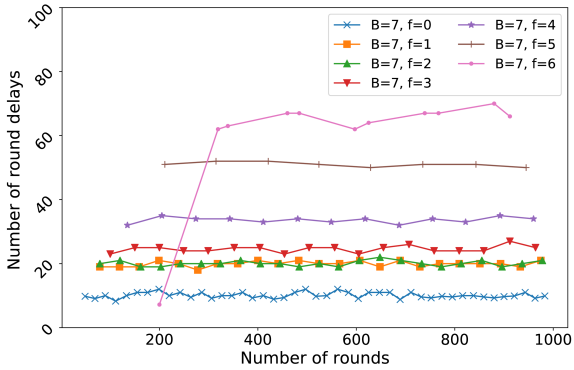
(b) DyPS with dynamic scaling of the SNPs set.

Figure 11. DyPS without and with dynamic scaling of the SNPs set for a GWAS consisting of 3,000 SNP positions ($LtoN(L) = 21,600$) ($B = 4, f = 0$).

We then measured the number of pending operations, and the overall number of applied genome operations. Figure 11a and Figure 11b respectively report those numbers for DyPS with or without dynamic scaling. DyPS without dynamic scaling was not able to apply any removal requests during the experiment. Moreover, at the end of the experiment, 14,119 genome addition and 3,572 genome removal requests were still pending. In total, 21,602 genomes have been added and none were removed. In contrast, DyPS could publish more safe releases starting from the first round. We ended the experiment at the 1250th round to compare the performance of both approaches. At that point DyPS was able to add 22,179 and to remove 7,580 genomes. At the end of the experiment, 19 genome additions and 2 genome removals remained, which represented a decrease of more than 99%



(a) Time to release in rounds for genome addition requests.



(b) Time to release in rounds for genome removal requests.

Figure 12. $(B - f)$ DyPS: Time to release in rounds for genome requests for a GWAS consisting of 300 SNP positions ($LtoN(L) = 1,598$) during 1,000 rounds, and different number of possibly colluding biocenters.

for both cases when compared to DyPS without dynamic scaling.

We also compared the CPU running time of the two versions. We noted that DyPS with dynamic scaling has a slight increase of CPU time when compared to DyPS without scaling, due to the fact that it needs additional analysis to dynamically evaluate and safely decide the increasing of the number of SNPs over which statistics are released. However it still remains practical and a magnitude lower than the *BF* approach for the selection of requests (which does not consider a dynamic scaling scheme).

5.5 Impact of colluding biocenters

In this section we evaluate DyPS' performance with colluding biocenters. For this experiment, we increase the number of biocenters to 7 and vary the number of colluding biocenters (f). This experiment was performed in the enclave. Figure 12 illustrates the pending rounds for addition and removal requests per update of the GWAS results for varying number f of colluding biocenters. As one could expect,

Table 5. Average number of processed addition and removal requests, number of GWAS releases, and average round delays for addition and removal requests depending on the number of colluding biocenters.

Threat Model (B, f)	*#Additions, #Removals, #Releases, #Add. round delays, #Rmv. round delays*
$B = 7, f = 0$	55,435 / 19,917 / 46 / 10.13 / 10.1
$B = 7, f = 1$	55,093 / 19,421 / 24 / 19.75 / 10.9
$B = 7, f = 2$	55,013 / 19,692 / 24 / 19.96 / 20.1
$B = 7, f = 3$	54,353 / 18,984 / 19 / 24.84 / 24.6
$B = 7, f = 4$	54,033 / 18,825 / 14 / 33.93 / 33.6
$B = 7, f = 5$	52,655 / 17,994 / 9 / 52.0 / 51.0
$B = 7, f = 6$	50,813 / 14,625 / 12 / 71.25 / 60.2

when more biocenters collude the requests are applied with more delay, since it takes more time to assemble larger sets of requests, which are required for safety when facing more adversaries. For example, with the first threat model ($f = 0$), the average number of pending rounds was 10.13 and 10.11 for addition and removal requests, respectively. On the other hand, with $f = 5$ and $f = 6$, the average processing delays were equal to 52 and 71.25 rounds for additions, and 53 and 60.2 for removals, respectively. An interesting event happens in Figure 12b for the $B = 7, f = 6$ line where there is a very small delay at the 200th round. It is explained by the fact that a release was created with the genomes of a single biocenter, in the 181st round (first release in Figure 12a), and then after just 19 rounds, this same biocenter was able to execute removal requests at the 200th round, which explains the short delay.

Table 5 details the number of genome additions and removals, the number of releases, and the average addition and removal request delays under several scenarios. As expected, fewer addition and removal operations are executed when facing stronger adversaries. However, the numbers decrease by at most 8.34% for the additions, and 26.57% for the removals when comparing without collusion ($f = 0$) to six colluding biocenters over seven ($f = 6$). The latter number is explained by the fact that DyPS does not currently wait to perform requests, and because removals require that sufficiently enough additions are simultaneously executed.

5.6 SNP selection for aggregate statistics

DyPS' releases of aggregate statistics depend on the SNP distribution among the set of added genomes. We first evaluate DyPS using the idash2017 dataset [33], which consists of 2,000 real genomes (1,000 case and 1,000 control). We used all the genomes in the control set as the adversary knowledge. For aggregate statistics, DyPS' release mechanisms require more extensive computations when the number of previous releases and when the number of genomes used per release

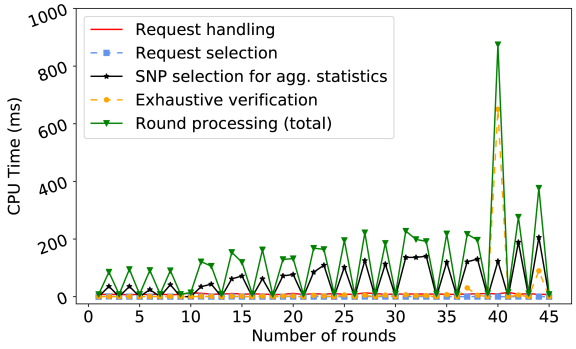


Figure 13. Running time for the different steps of DyPS execution for a GWAS studying 10 SNP positions ($B = 5, f = 0$). Reference group size: 1,000. Total number of real genomes used: 2,000.

increase (cf. Section 4.4). We therefore primarily consider scenarios where $f = 0$, because it maximizes the number of releases and their size. In this section, we consider a GWAS that studies a small set of L SNPs (i.e., the top 10 most significant SNPs) so that releases are more frequent. Given these parameter values, we study the worst case of the exhaustive verification procedure.

Figure 13 details DyPS' CPU consumption during each round per category: request handling, request selection, SNP selection for aggregate statistic release, exhaustive verification, and total round processing. Each CPU running time peak happened when a round resulted in a release of GWAS statistics. DyPS' selection algorithm, which runs for every round, used less than 200 ms for almost each round, and less than 1 second overall, which is very reasonable given that sequencing a genome usually requires around 1 day. For the largest measured value, obtained in the 42nd round, DyPS verified more than 2,359,296 combinations of releases in the enclave. The largest part of the computation was used by the exhaustive verification process for aggregate statistics release.

5.7 DyPS vs. static release of aggregate statistics

Next, we compare DyPS with a state-of-the-art static release algorithm over a larger set (the top 1,000 SNPs) of the idash2017 dataset in Figure 14. More precisely, we measure over how many SNPs both approaches are able to release aggregate statistics only, which enables frequent updates. We use the LR-metric [11] to decide whether singlewise frequencies can be released. The static release method identified that singlewise frequencies could be safely released over 45 SNPs out of the 1,000 studied, using the full set of genomes remaining at the end of the experiment (i.e., in only one release - the dashed bar). On the other hand, the number of selected SNPs for DyPS varied according to the rounds, as expected,

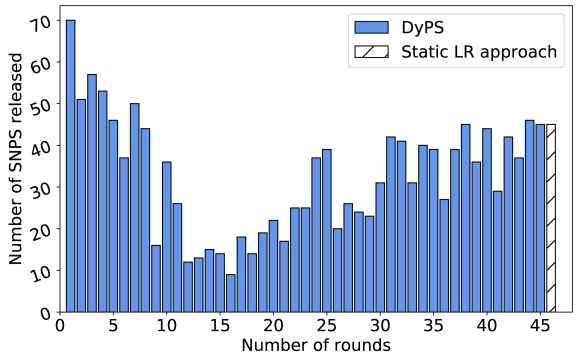


Figure 14. Comparison between a static approach and DyPS for releases of aggregate statistics for a GWAS studying 1,000 SNP positions ($B = 5$, $f = 0$). Reference group size: 1,000. Total number of real genomes used: 2,000.

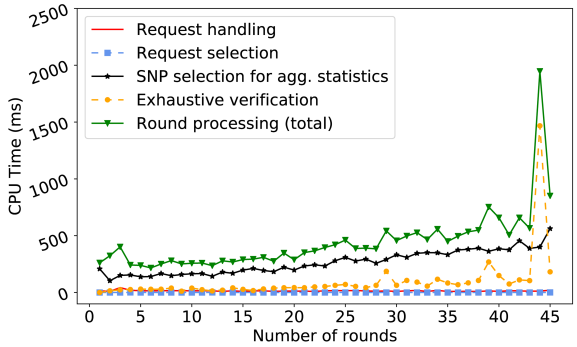


Figure 15. Running time for the different steps of DyPS for a GWAS studying 1,000 SNP positions ($B = 5$, $f = 0$). Reference group size: 1,000. Total number of real genomes used: 2,000.

that is, in each different round, a different distribution of genomes participated in a release and, therefore, the set of SNPs over which statistics can be released evolves. The maximum number of SNPs DyPS released during the experiment was 70, while small releases, which do not prevent previous releases to be accessed, were more frequent. Note that in this experiment, DyPS released aggregate statistics after every round, while test statistics would not have been updated as frequently. Therefore, one can conclude that DyPS not only provides more frequent releases but also over a largest set of SNP positions. In a similar scenario with 350 SNPs, DyPS was able to release 2.6 times more statistics (i.e., multiple safe releases with at most 44 SNPs instead of 17 only released once).

Next, we provide the results for the CPU running time in Figure 15. As can be noted, in the 44th round a more extensive verification took place, which has checked 4,972,331 combinations of releases in total. The running time was 1,467

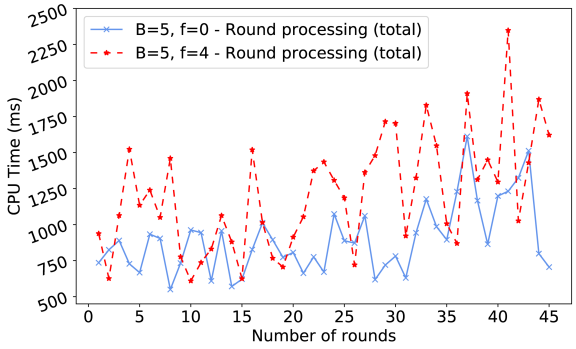


Figure 16. Running time for $f = 0$ and $f = 4$ with aggregate statistics computed over 1,000 SNP positions. Reference group size: 1,000. Total number of real genomes used: 2,000.

milliseconds. Furthermore, comparing to Figure 13 experiment with 10 SNPs, there is now an expected slight increase in the SNP selection software algorithm running time (star marker line) because now a larger number of SNPs have been checked. Nevertheless, even increasing the SNP set size by 100x times (from 10 to 1,000 SNPs scenario, the longest time have just approximately doubled in average (200 to 500 ms). Similarly, the longest time for the exhaustive verification, took approximately 2,000 ms in the 42nd round.

We also evaluated the impact of colluding biocenters on aggregate statistics. When facing colluding biocenters, DyPS has also to evaluate combinations of genomes. This results in an increased running time due to the additional verifications. Considering the same scenario, and now using f equals to $(B-1)$, we noted that DyPS's running time increases slightly. We show in Figure 16 the computation time required by DyPS when the number of colluding biocenters is either 0 or $B-1$, where $B = 5$. In particular, in this new experiment the peak and average running time for $f = 0$ were 1,612 and 896 milliseconds, respectively. Whereas for the $f = 4$ case, it took 2,348 and 1,212 milliseconds.

5.8 DyPS over a large-scale GWAS

We now consider another real genome dataset, dbGAP *phs001039.v1.p1* [6], which consists of more than 35,000 genomes from which we could use 27,895 under the General Research Use (GRU) consent, of which 14,860 genomes are cases and 13,035 are controls, respectively. We study the SNP positions that appear in both cohorts. We consider chromosome 1 as it is the chromosome with the largest number of remaining SNPs. We consider 5,000 SNP positions to evaluate DyPS's algorithm over this larger dataset. Besides, we multiplied both the addition and removal lambda parameters by 16 ($\lambda = 128$ for additions and $\lambda = 96$ for removals), so that more genome operations are generated per round. We considered the whole control dataset (13,035 genomes) as the adversary reference group.

In Figure 17, we show the performance of DyPS over this larger dataset. Overall, 12,418 genomes were added, and 5,120 genomes were removed. Compared to the experiment in Figure 15, this experiment had an expected longer running time (average of 207 seconds, and peak of 2,500 seconds, when a cohort made of more than 27,800 genomes was studied) for the SNP selection algorithm because of the larger genomes cohort (i.e., more participating genomes and a larger reference group). On the other hand, Figure 17 shows a shorter running time for the exhaustive verification step due to a smaller number of SNP positions over which statistics could be released. This was because most of the SNP positions were being filtered out by the linkage disequilibrium and MAF step of DyPS over previous rounds.

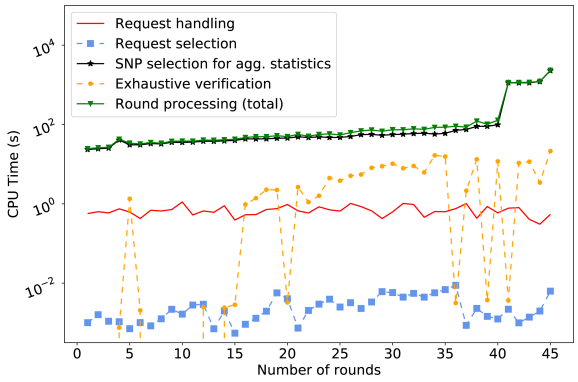


Figure 17. Running time for the different steps of DyPS execution for a GWAS studying 5,000 SNP positions ($B = 5, f = 0$). Reference group size: 13,035. Total number of real genomes used: 27,895.

In addition, in scenarios where DyPS deals with a larger number of SNPs, it could rely on existing SNPs batching mechanisms inside enclaves, such as [34], in which SNPs are firstly separated in batches of equal size, and processed separately. Later, SNPs in different batches are processed in a crossed-over manner in order to keep a global set of safe SNPs updated as batches are processed.

Table 6. DyPS' average memory consumption inside the enclave depending on several controls group sizes and 5,000 SNPs.

(#B / #f) DyPS inside enclaves	Average Memory Consumption (KB)
($B = 5/f = 0$), controls: 1,000	2,160
($B = 5/f = 0$), controls: 5,000	2,224
($B = 5/f = 0$), controls: 7,500	2,228
($B = 5/f = 0$), controls: 10,000	2,228
($B = 5/f = 0$), controls: 13,035	2,228

We report DyPS' memory consumption in Table 6 when different sizes of control groups (i.e., adversary reference groups) are used. The memory consumption of the machine was identical to the one reported in previous experiments, i.e., around 2 MB. DyPS can therefore support a large number of genomes and SNP positions without significant performance penalty. Overall, the memory consumption stays well below the theoretical 128 MB (of which only 96 MB is usable without paging [31]) memory limitation of SGX enclaves.

6 Related work

Defenses against membership attacks. Several works proposed solutions against membership attacks for static datasets, while DyPS targets dynamic datasets. DyPS builds on SecureGenome [11] for singlewise frequencies, and extends it to pairwise frequencies in its SNP selection algorithm (using the T_r [3] and/or Λ [12] metrics), to determine the set of SNPs over which frequencies can be safely released. Craig et al. [35] advocated for a similar approach based on the Positive Predictive Value (PPV). Im et al. [9] showed that the membership attack power increases when regression coefficients from quantitative multi-phenotype are available. Fernandes et al. [36] considered separating a genome over multiple sensitivity levels to prevent reidentification attacks.

Privacy-preserving GWAS releases.

Zhou et al. [12] evaluated the theoretical complexity of attacks on GWAS results. DyPS further builds on their results by offering a dynamic and federated architecture. Data anonymization methods enforce privacy properties such as t-closeness, k-anonymity and l-diversity [21]. However, k-anonymous genomic datasets have been deanonymized [37–39], even with high dimensional data [40]. Differential Privacy (DP) [41], which adds noise to the results, has also been used [42–48]. In comparison, DyPS leaves the data unmodified. Jiang et al. [47] proposed a privacy-budget approach based on the *LR*-test that balances data perturbation with privacy risks. Simmon et al. [46] propose two DP frameworks for GWAS: PrivSTRAT applies data perturbation considering the group stratification in a study and PrivLMM is based on Linear Mixed Models (LLMs). Johnson et al. [43] offer a DP approach for GWAS based on adding noise to data based on their shifting method that considers distance scores. Lu et al. [48] described a Distributed Differential Privacy (DDP), in which parties perturb their local genomic data shares before sharing them. Similarly, another noise-based approach [39] combines the addition of minimal amounts of noise perturbation with Bayesian and Markov Chain Monte Carlo techniques. Nevertheless, given that the privacy level certainty sustained by DP depends on the statistical independence of the records in a dataset, previous works have shown that highly correlated records in datasets can diminish the DP's guarantees [49, 50]. In addition, if not well designed,

DP approaches can suffer from collusion attacks, which can disclose individuals' data [51]. DP under continual observation [52, 53] and for growing databases [54] seems promising for the dynamic releases of GWAS results. To the best of our knowledge, no work uses such approaches for GWAS. One limitation might be that the noise added to the data increases with the number of colluding players and the number of releases. Studying whether DyPS can leverage those properties is left for future work.

Privacy-preserving GWAS computations.

Several works rely on homomorphic encryption (HE), secure multi-party computation (MPC) or trusted execution environments (TEE) for GWAS computations. These works could benefit from DyPS to dynamically decide when computations should take place, and their results be released. Lu et al. [16] describe a method where a central entity generates HE keys that biocenters used to encrypt their data, so that a public cloud can use homomorphic computations to obtain GWAS statistics. Kim et al. [17] proposed a fully HE scheme to run χ^2 tests using 80-bit key security. More recently, Bonte et al. [55] introduced a distributed GWAS system, which uses HE and MPC methods, and answers *yes/no* responses for putative markers SNPs (rather than releasing χ^2 values). More recently, yet another HE-based approach [56] offers complementary mechanisms to accelerate and increase the performance of GWAS analysis, such as parallelization and encoding techniques. Kamm et al. [57] a secure MPC framework where institutes share their genome dataset to a third-party data storage for χ^2 tests computing. Zhang et al. [23] use secret sharing that assumes (n, t)-threshold for corrupted parties. [21] performs χ^2 and p-tests in a privacy-preserving way, and [22] performs privacy-preserving χ^2 and MAF processing. However, these two approaches are restricted to two parties, and can not be applied as is in a federated system. Recently, [20] also proposed MPC techniques for privacy-preserving GWAS based on population biases, however, they have not offered any mechanisms to avoid attacks on aggregate released results. Raisaro et al. [14] offer a decentralized and privacy-preserving data sharing of genomic data to answer health record queries and build on Unlynx [58], which incorporates known privacy-preserving techniques, such as HE and DP. PREMIX [59] relied on Intel SGX enclaves to evaluate individual genomic admixture. SAFETY, proposed by Sadat et al. [60], combines HE for the summation of participant's inputs, and SGX enclaves for the more heavy statistical processing. In addition, PRINCESS [34] also performs GWAS tests using SGX enclaves for rare-disease collaboration studies. PRESAGE [61] applies encoding and indexing methods on genomic data to answer private queries with high performance. After encryption, the data is outsourced to an untrusted party, e.g., an SGX-enabled cloud provider, which answers genomic queries with encrypted results.

DyPS is the first secure federated platform for genomic statistics (i.e., GWAS) which addresses the problem of dynamic computation and release of GWAS, both on a dynamic supply of the number of considered genomes in the study, and a change of users consent. To the best of our knowledge, DyPS is the first work which copes with both aspects of the problem: (i) respecting conditions for a safe release of GWAS (i.e., ensuring privacy of participants), and (ii) securing the federated computation of GWAS (i.e., do not leak information belonging to one biocenter to others). These two aspects have been addressed independently in the literature in works that either prevent secure federated computation, or do not preserve the privacy of participants of the study.

SGX-based privacy-preserving systems.

TEEs and in particular Intel SGX have been extensively used to build secure and privacy-preserving systems. DyPS illustrates the benefits of this technology for performance-sensitive genomic data processing applications [62]. Several side-channel attacks affecting Intel SGX have been described, e.g., timing attacks from page faults, or memory access patterns [30]. Several approaches have been proposed to mitigate these attacks, including memory oblivious algorithms. For instance, Mandal et al. [63] offer oblivious approaches for secure genomic data analysis using TEE enclaves. One can also limit the page memory access patterns to 4 kB to circumvent the side-channel attacks [34, 61]. Studying the impact of memory-oblivious mechanisms on DyPS' performance is future work, and will require removing the assumption that the pseudonyms of the genomes used in a statistics release are known.

7 Conclusion

In this paper, we presented DyPS, a novel framework to conduct federated genomic data analysis that enables both secure computation and release of GWAS results even when facing collusion among participating dataholders. In addition, DyPS enables dynamic releases of GWAS results according to the evolution of the considered number of genomic variations, individuals, or involved biocenters while enforcing privacy. To further speed up the release of the results, DyPS uses a scaling mechanism that progressively increases the number of considered genomic variations.

DyPS leverages an Intel SGX enclave to secure the data of each biocenter. We extensively evaluated its performance considering up to 300,000 SNPs and more than 6 million simulated genomes, and using two datasets made of 2,000 and 35,000 real genomes, respectively. We successfully demonstrated the practicability of both test and aggregate statistics release mechanisms, and compared them against baseline approaches. For test statistics, we compared DyPS to an approach that would immediately update the results whenever a sufficiently large batch of requests could be assembled. This

latter approach put up to 8% of the genomes at risk, while DyPS prevents all privacy leaks. For aggregate statistics, we considered a static approach that would wait for all requests to have been received before releasing statistics. Compared to this approach, DyPS is able to provide earlier aggregate statistics releases over the same number of SNPs. These earlier releases occasionally provided statistics including up to 2.6 times as many SNPs. Future work includes extending DyPS to tolerate malicious adversaries, and considering memory-oblivious algorithms.

Acknowledgments. This work was partially supported by the European Union under the H2020 Programme Grant Agreement No. 830929 (CyberSec4Europe) and by the Fonds National de la Recherche Luxembourg through PEARL grant FNR/P14/8149128.

References

[1] Tanya Lewis. *Human Genome Project Marks 10th Anniversary.* https://www.livescience.com/28708-human-genome-project-anniversary.html. Accessed on: January 7th, 2019.

[2] Robert Sladek, Ghislain Rocheleau, Johan Rung, Christian Dina, Lishuang Shen, David Serre, Philippe Boutin, Daniel Vincent, Alexandre Belisle, Samy Hadjadj, et al. A genome-wide association study identifies novel risk loci for type 2 diabetes. *Nature*, 445(7130):881, 2007.

[3] Rui Wang, Yong Fuga Li, XiaoFeng Wang, Haixu Tang, and Xiaoyong Zhou. Learning your identity and disease from research papers: Information leaks in genome wide association study. In *CCS*, 2009.

[4] Ruichu Cai, Zhifeng Hao, Marianne Winslett, Xiaokui Xiao, Yin Yang, Zhenjie Zhang, and Shuigeng Zhou. Deterministic identification of specific individuals from gwas results. *Bioinformatics*, 31(11):1701–1707, 2015.

[5] Laura L Rodriguez, Lisa D Brooks, Judith H Greenberg, and Eric D Green. The complexities of genomic identifiability. *Science*, 339(6117):275–276, 2013.

[6] Lorelei Walker, Helene Starks, Kathleen M West, and Stephanie M Fullerton. dbgap data access requests: a call for greater transparency. *Science translational medicine*, 3(113):113–134, 2011.

[7] Nils Homer, Szabolcs Szelinger, Margot Redman, David Duggan, Waibhav Tembe, Jill Muehling, John V Pearson, Dietrich A Stephan, Stanley F Nelson, and David W Craig. Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLoS genetics*, 4(8), 2008.

[8] Kevin B Jacobs, Meredith Yeager, Sholom Wacholder, David Craig, Peter Kraft, David J Hunter, Justin Paschal, Teri A Manolio, Margaret Tucker, Robert N Hoover, et al. A new statistic and its power to infer membership in a genome-wide association study using genotype frequencies. *Nature genetics*, 41(11):1253, 2009.

[9] Hae Kyung Im, Eric R Gamazon, Dan L Nicolae, and Nancy J Cox. On sharing quantitative trait gwas results in an era of multiple-omics data and the limits of genomic privacy. *The American Journal of Human Genetics*, 90(4):591–598, 2012.

[10] Elias A Zerhouni and Elizabeth G Nabel. Protecting aggregate genomic data. *Science*, 322(5898):44–44, 2008.

[11] Sriram Sankararaman, Guillaume Obozinski, Michael I Jordan, and Eran Halperin. Genomic privacy and limits of individual detection in a pool. *Nature genetics*, 41(9):965–967, 2009.

[12] Xiaoyong Zhou, Bo Peng, Yong Fuga Li, Yangyi Chen, Haixu Tang, and XiaoFeng Wang. To release or not to release: Evaluating information leaks in aggregate human-genome data. In *Esorics*, 2011.

[13] Paulo Esteves Verissimo and Alysson Bessani. E-biobanking: What have you done to my cell samples? *Security & Privacy*, 11(6):62–65, 2013.

[14] Jean Louis Raisaro, Juan Ramón Troncoso-Pastoriza, Mickaël Misbach, E Sousa Gomes de Sá, Joao André, Sylvain Pradervand, Edoardo Missiaglia, Olivier Michielin, Bryan Alexander Ford, and Jean-Pierre Hubaux. Medco: Enabling privacy-conscious exploration of distributed clinical and genomic data. In *GenoPri*, 2017.

[15] Arun Iyengar, Ashish Kundu, Upendra Sharma, and Ping Zhang. A trusted healthcare data analytics cloud platform. In *ICDCS*, 2018.

[16] Wen-Jie Lu, Yoshiji Yamada, and Jun Sakuma. Privacy-preserving genome-wide association studies on cloud environment using fully homomorphic encryption. *BMC medical informatics and decision making*, 15(5):S1, 2015.

[17] Miran Kim and Kristin Lauter. Private genome analysis through homomorphic encryption. *BMC medical informatics and decision making*, 15(5):S3, 2015.

[18] Yuchen Zhang, Wenrui Dai, Xiaoqian Jiang, Hongkai Xiong, and Shuang Wang. Foresee: Fully outsourced secure genome study based on homomorphic encryption. 15(5):S5, 2015.

[19] Marcelo Blatt, Alexander Gusev, Yuriy Polyakov, and Shafi Goldwasser. Secure large-scale genome-wide association studies using homomorphic encryption. *National Academy of Sciences*, 117(21):11608–11613, 2020.

[20] Hyunghoon Cho, David J Wu, and Bonnie Berger. Secure genome-wide association analysis using multiparty computation. *Nature biotechnology*, 36(6):547, 2018.

[21] Oleksandr Tkachenko, Christian Weinert, Thomas Schneider, and Kay Hamacher. Large-scale privacy-preserving statistical computations for distributed genome-wide association studies. In *Asia CCS*, 2018.

[22] Scott D Constable, Yuzhe Tang, Shuang Wang, Xiaoqian Jiang, and Steve Chapin. Privacy-preserving gwas analysis on federated genomic datasets. *BMC medical informatics and decision making*, 15(5):S2, 2015.

[23] Yihua Zhang, Marina Blanton, and Ghada Almashaqbeh. Secure distributed genome analysis for gwas and sequence comparison computation. *BMC medical informatics and decision making*, 15(5):S4, 2015.

[24] Fida K Dankar, Marton Gergely, Bradley Malin, Radja Badji, Samar K Dankar, and Khaled Shuaib. Dynamic-informed consent: A potential solution for ethical dilemmas in population sequencing initiatives. *Computational and Structural Biotechnology Journal*, 2020.

[25] Jérémie Decouchant, Maria Fernandes, Marcus Völp, Francisco M Couto, and Paulo Esteves-Verissimo. Accurate filtering of privacy-sensitive information in raw genomic data. *Journal of biomedical informatics*, 82:1–12, 2018.

[26] Mathias Humbert, Erman Ayday, Jean-Pierre Hubaux, and Amalio Telenti. Quantifying interdependent risks in genomic privacy. *TOPS*, 20(1):3, 2017.

[27] Zhicong Huang, Erman Ayday, Jacques Fellay, Jean-Pierre Hubaux, and Ari Juels. Genoguard: Protecting genomic data against brute-force attacks. In *Security & Privacy*, 2015.

[28] Jean Louis Raisaro, Carmela Troncoso, Mathias Humbert, Zoltan Kutalik, Amalio Telenti, and Jean-Pierre Hubaux. Genoshare: Supporting privacy-informed decisions for sharing exact genomic data. Technical report, EPFL infoscience, 2017.

[29] Peter Ney, Karl Koscher, Lee Organick, Luis Ceze, and Tadayoshi Kohno. Computer security, privacy, and dna sequencing: Compromising computers with synthesized dna, privacy leaks, and more. In *USENIX*, 2017.

[30] Ferdinand Brasser, Urs Müller, Alexandra Dmitrienko, Kari Kostiainen, Srdjan Capkun, and Ahmad-Reza Sadeghi. Software grand exposure: $\{SGX\}$ cache attacks are practical. In *WOOT*, 2017.

[31] Victor Costan and Srinivas Devadas. Intel sgx explained. *IACR Cryptology ePrint Archive*, 2016(086):1–118, 2016.

[32] Chia-Che Tsai, Donald E Porter, and Mona Vij. Graphene-sgx: A practical library os for unmodified applications on sgx. In *USENIX ATC*, 2017.

[33] *iDASH Privacy & Security Challenge - secure genome analysis competition.* http://www.humangenomeprivacy.org/2017/competition-tasks.html. Accessed on: March 13rd, 2020.

[34] Feng Chen, Shuang Wang, Xiaoqian Jiang, Sijie Ding, Yao Lu, Jihoon Kim, S Cenk Sahinalp, Chisato Shimizu, Jane C Burns, Victoria J Wright, et al. Princess: Privacy-protecting rare disease international network collaboration via encryption through software guard extensions. *Bioinformatics*, 33(6):871–878, 2016.

[35] David W Craig, Robert Goor, Zhenyan Wang, Justin Paschall, Jim Ostell, Mike Feolo, Stephen T Sherry, and Teri A Manolio. Assessing and managing risk when sharing aggregate genetic variant data. *Nature reviews Genetics*, 12(10):730, 2011.

[36] Maria Fernandes, Jérémie Decouchant, Marcus Völp, Francisco M Couto, and Paulo Esteves-Verissimo. Dna-seal: Sensitivity levels to optimize the performance of privacy-preserving dna alignment. *IEEE Journal of Biomedical and Health Informatics*, 24(3):907–915, 2019.

[37] Latanya Sweeney, Akua Abu, and Julia Winn. Identifying participants in the personal genome project by name (a re-identification experiment). *arXiv preprint:1304.7605*, 2013.

[38] Jaideep Vaidya, Basit Shafiq, Xiaoqian Jiang, and Lucila Ohno-Machado. Identifying inference attacks against healthcare data repositories. *AMIA Summits on Translational Science Proceedings*, 2013:262, 2013.

[39] Sean Simmons, Bonnie Berger, and Cenk S Sahinalp. Protecting genomic data privacy with probabilistic modeling. In *PSB*, 2019.

[40] Guy Zyskind, Oz Nathan, et al. Decentralizing privacy: Using blockchain to protect personal data. In *SPW*, 2015.

[41] C Dwork. *Differential privacy*. Springer, 2011.

[42] Caroline Uhlerop, Aleksandra Slavković, and Stephen E Fienberg. Privacy-preserving data sharing for genome-wide association studies. *The Journal of privacy and confidentiality*, 5(1):137, 2013.

[43] Aaron Johnson and Vitaly Shmatikov. Privacy-preserving data exploration in genome-wide association studies. In *SIGKDD*, pages 1079–1087, 2013.

[44] Yongan Zhao, Xiaofeng Wang, Xiaoqian Jiang, Lucila Ohno-Machado, and Haixu Tang. Choosing blindly but wisely: differentially private solicitation of dna datasets for disease marker discovery. *Journal of the American Medical Informatics Association*, 22(1):100–108, 2014.

[45] Florian Tramèr, Zhicong Huang, Jean-Pierre Hubaux, and Erman Ayday. Differential privacy with bounded priors: reconciling utility and privacy in genome-wide association studies. In *SIGSAC*, pages 1286–1297, 2015.

[46] Sean Simmons, Cenk Sahinalp, and Bonnie Berger. Enabling privacy-preserving gwass in heterogeneous human populations. *Cell systems*, 3(1):54–61, 2016.

[47] Xiaoqian Jiang, Yongan Zhao, Xiaofeng Wang, Bradley Malin, Shuang Wang, Lucila Ohno-Machado, and Haixu Tang. A community assessment of privacy preserving techniques for human genomes. *BMC medical informatics and decision making*, 14(1):S1, 2014.

[48] Zhigang Lu and Hong Shen. A new lower bound of privacy budget for distributed differential privacy. In *PDCAT*, pages 25–32, 2017.

[49] Daniel Kifer and Ashwin Machanavajjhala. No free lunch in data privacy. In *MOD*, 2011.

[50] Changchang Liu, Supriyo Chakraborty, and Prateek Mittal. Dependence makes you vulnberable: Differential privacy under dependent tuples. In *NDSS*, 2016.

[51] Fabienne Eigner, Aniket Kate, Matteo Maffei, Francesca Pampaloni, and Ivan Pryvalov. Differentially private data aggregation with optimal utility. In *ACSAC*, 2014.

[52] Cynthia Dwork, Moni Naor, Toniann Pitassi, and Guy N Rothblum. Differential privacy under continual observation. In *STOC*, 2010.

[53] T-H Hubert Chan, Elaine Shi, and Dawn Song. Private and continual release of statistics. *TISSEC*, 14(3):1–24, 2011.

[54] Rachel Cummings, Sara Krehbiel, Kevin A Lai, and Uthaipon Tantipongpipat. Differential privacy for growing databases. In *Advances in Neural Information Processing Systems*, 2018.

[55] Charlotte Bonte, Eleftheria Makri, Amin Ardeshirdavani, Jaak Simm, Yves Moreau, and Frederik Vercauteren. Towards practical privacy-preserving genome-wide association study. *BMC bioinformatics*, 19(1):537, 2018.

[56] Marcelo Blatt, Alexander Gusev, Yuriy Polyakov, and Shafi Goldwasser. Secure large-scale genome-wide association studies using homomorphic encryption. *National Academy of Sciences*, 2020.

[57] Liina Kamm, Dan Bogdanov, Sven Laur, and Jaak Vilo. A new way to protect privacy in large-scale genome-wide association studies. *Bioinformatics*, 29(7):886–893, 2013.

[58] David Froelicher, Patricia Egger, João Sá Sousa, Jean Louis Raisaro, Zhicong Huang, Christian Mouchet, Bryan Ford, and Jean-Pierre Hubaux. Unlynx: a decentralized system for privacy-conscious data sharing. *PETS*, 2017(4):232–250, 2017.

[59] Feng Chen, Michelle Dow, Sijie Ding, Yao Lu, Xiaoqian Jiang, Hua Tang, and Shuang Wang. Premix: Privacy-preserving estimation of individual admixture. *AMIA Annual Symposium*, 2016:1747, 2016.

[60] Md Nazmus Sadat, Md Momin Al Aziz, Noman Mohammed, Feng Chen, Xiaoqian Jiang, and Shuang Wang. Safety: secure gwas in federated environment through a hybrid solution. *TCBB*, 16(1):93–102, 2018.

[61] Feng Chen, Chenghong Wang, Wenrui Dai, Xiaoqian Jiang, Noman Mohammed, Md Momin Al Aziz, Md Nazmus Sadat, Cenk Sahinalp, Kristin Lauter, and Shuang Wang. Presage: Privacy-preserving genetic testing via software guard extension. *BMC medical genomics*, 10(2):48, 2017.

[62] Christoph Lambert, Maria Fernandes, Jérémie Decouchant, and Paulo Esteves-Verissimo. Maskal: Privacy preserving masked reads alignment using intel sgx. In *SRDS*, 2018.

[63] Avradip Mandal, John C Mitchell, Hart Montgomery, and Arnab Roy. Data oblivious genome variants search on intel sgx. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*. Springer, 2018.

A Requests selection for test statistics

A.1 Composition property and proofs

We now demonstrate that the method we developed to select the set of genomes to be added, or removed, from a GWAS release prevents both membership attacks on test statistics, and recovery attacks on both aggregate and test statistics. We refer the reader to Table 4 for a summary of the equations we rely on to evaluate the conditions of feasibility of those attacks. More precisely, those equations state that there is a function $NtoL(L)$ such that a GWAS release that studies L SNPs can be considered secure if it used $LtoN(L)$ genomes. Similarly, one could invert these equations to discover how many SNPs L can be safely released according to the number of genomes N, namely $NtoL(N)$.

In the following, let A_i be the set of genome additions and R_i the set of genome removals used for a specific release, or update, i of the GWAS results.

Theorem 1 (Vertical expansions with $f = 0$)**.** *If each release i is such that $|A_i| + |R_i| \geq LtoN(L)$ and that $|A_i| \geq |R_i|$, then each combination of releases involves more than $LtoN(L)$*

genomes, which prevents an adversary to successfully launch a privacy attack.

Proof. We prove this theorem by induction. The first release does not contain any genome removal and, therefore, adds more than $LtoN(L)$ genomes. The property to prove is then verified for the first release. Let us assume that this property is verified for any combination of releases whose ids are lower than or equal to i. Let j be the ID of the $(i+1)$-th release, which contains the additions and removals of genomes A_j and R_j. Let us consider a combination of releases whose ids are lower than or equal to i. This combination contains a set of genome additions A and a set of genome removals R. We now show that if we were to combine this combination with release j we would still obtain a secure release.

The number of genomes that an adversary can isolate by combining the releases is equal to $|R|+|A\setminus R_j|+|R_j\setminus A|+|A_j|$. By adding and removing $|A\cap R_j|$ to this expression one can obtain:

$\big(|R|+|A\setminus R_j|+|A\cap R_j|\big)+\big(-|A\cap R_j|+|R_j\setminus A|+|A_j|\big)$ (1).

The values of the two parts of the previous expression can be bounded thanks to the following two inequalities:

- By assumption, we have $|A|+|R|\geq LtoN(L)$ and, therefore, $|R|+|A\setminus R_j|+|A\cap R_j|\geq LtoN(L)$ (2)
- By construction, $|A_j|\geq|R_j|$, and therefore $|A_j|\geq|A\cap R_j|+|R_j\setminus A|$ (3)

We can now bound each term in (1). First, we already established (in (2)) that $(|R|+|A\setminus R_j|+|A\cap R_j|)=|A|+|R|\geq LtoN(L)$, which bounds the first term in (1). Second, by using (3), one can see that $-|A\cap R_j|+|R_j\setminus A|+|A_j|\geq -|A\cap R_j|+|R_j\setminus A|+\big(|A\cap R_j|+|R_j\setminus A|\big)\geq 2*|R_j\setminus A|\geq 0$, which bounds the second term in (1). By adding these two inequalities, we finally obtain that $|R|+|A\setminus R_j|+|R_j\setminus A|+|A_j|\geq LtoN(L)$, which concludes. □

Theorem 2 (Horizontal expansions with $f=0$)**.** *Let (A_i,R_i) be the set of genome additions and removals respectively executed during the GWAS results update i. The horizontal expansion algorithm allows an expansion of L SNPs and does not allow an adversary to successfully launch a privacy attack.*

Proof. By construction, the released set of L_{i+1} SNPs have been chosen so that $|A_i|+|R_i|\geq NtoL(L_{i+1})$, which prevents any genomic data over the SNPs that are newly considered in release $i+1$ to be inferred. □

Theorem 3 (Diagonal expansions with $f=0$)**.** *Let (A_i,R_i) and (A_{i+1},R_{i+1}) be the sets of genome additions and removals respectively executed during the GWAS results updates i and $i+1$, between which a diagonal expansion occurred. The diagonal expansion algorithm does not allow an adversary to successfully launch a privacy attack.*

Proof. A diagonal expansion can be seen as a combination of vertical and horizontal updates, which are respectively proven safe in Theorems 1 and 2. □

In the following, let $A_{i,S}$ be the set of genome additions and let $R_{i,S}$ the set of genome removals used for a specific release, or update, i of the GWAS results by a set S of players (i.e., biocenters). We now show that the previous results applied to the situation where up to f of the B biocenters might be colluding.

Theorem 4 (Diagonal expansions with $f\neq 0$)**.** *For any set S of $(B-f)$ non-colluding biocenters, if each release i either verifies:*

- $|A_{i,S}|+|R_{i,S}|\geq LtoN(L)$ *and* $|A_{i,S}|\geq|R_{i,S}|$, *or*
- $|A_{i,S}|=0$ *and* $|R_{i,S}|=0$,

then each combination of releases involves either no genomes from the $(B-f)$ biocenters at all, or more than $LtoN(L)$ genomes, which prevents an adversary to successfully launch a privacy attack.

Proof. A release that does not contain any additions or removals cannot leak any private information. We can therefore only reason about combinations of releases that each satisfy the first condition we listed. This Theorem is therefore a direct consequence of Theorem 3, if one considers the genomes that have been released by a given subset of $(B-f)$ biocenters. □

A.2 Pseudocode

DyPS uses the algorithm reported in Figure 18 to select the biocenters that participate in a batch of requests to be executed. In the case of test statistics, all addition requests from a selected biocenter are selected, and a lower or equal number of removals. This algorithm assumes that up to f biocenters are colluding, with $f\leq(B-1)$.

DyPS first retrieves and binds the requests to their corresponding biocenters in FIFO order (line 7 to 9), and adds them to `bioList`. After gathering the requests from the biocenters, DyPS sorts the list of biocenters according to their number of addition requests (line 10), before selecting a batch of requests (lines 11 to 34).

The rationale behind the selection algorithm is to select a group of biocenters such that their combined requests cannot be attacked by the f biocenters that participate the most, and who might be colluding. From line 12 to 20, DyPS checks if the number of additions of the i selected biocenters are large enough considering the requests of f malicious biocenters and that this number is equal or greater than $LtoN(L)$.

If such a set of biocenters is not found, DyPS checks if some biocenters have enough requests to update statistics individually, considering Theorem 2 in the previous section and $LtoN(L)$ (lines 21 to 28). Note that during this step, the algorithm limits the number of removals per biocenter to the number of additions it is also executing. From lines 30 to 33, DyPS checks if biocenters were selected and adds them to the list of selected biocenters (`selectedBios`). Finally, the algorithm retrieves the selected biocenters and returns the

```
1:  procedure DyPS request selection algorithm(B, f, L)
2:      Input: B set of biocenters, f number of colluding players, L number of SNPs.
3:      Output: sets of selected genome addition and removal requests.
4:      Uses: NtoL(L), the minimum number of genomes required to update L SNPs.
5:      bioList = ∅;
6:      selectedBios = ∅
7:      for b in B do// retrieve pending requests from each biocenter in FIFO order
8:          bioList[b.id].add(b.toAddGenomes,b.toRmvGenomes);
9:      end for
10:     bioList.sort();// sort using the number of addition requests
11:     istart = −1;
12:     for (int i = 0; i < B; i++) do
13:         if (bioList[i].addCount == 0) then
14:             continue;
15:         end if
16:         if (bioList.size() - i > f and sumBioReq(bioList, i, bioList.size() − f − 1) ≥ LtoN(L) then
17:             istart = i;
18:             break;
19:         end if
20:     end for
21:     if (istart == −1) then // assemble all biocenters that individually have enough operations
22:         for (int i = 0; i < B; i++) do
23:             if (bioList[i].addCount ≥ bioList[i].rmvCount and bioList[i].addCount + bioList[i].rmvCount ≥ LtoN(L)) then
24:                 istart = i;
25:                 break;
26:             end if
27:         end for
28:     end if
29:     if (istart != −1) then // assemble the selected biocenters
30:         for (int i = istart; i < B; i++) do
31:             selectedBios.add(bioList[i]);
32:         end for
33:     end if
34:     // assemble the batch of requests from selected biocenters
35:     for (int i = 0; i < selectedBios.size(); i++) do
36:         Adds_Batch := selectedBios[i].addRequests
37:         Rms_Batch := selectedBios[i].rmvRequests
38:     end for
39: end procedure
40: // release test statistic over selected set of requests
41: computeTestStatistics(Adds_Batch, Rms_Batch)
```

Figure 18. DyPS pseudocode for requests selection and test statistic releases.

sets of addition and removal requests that can be executed to update the GWAS test statistics (line 35 to 39). In the case of aggregate statistics, the actual composition of the requests batch is determined by the dedicated SNPs selection algorithm (see Appendix B).

B SNPs selection for aggregate statistics

B.1 Verifications for singlewise and pairwise statistics

In order to extend the mechanism that finds the list of SNPs over which statistics can be released using a batch of genome operations from singlewise frequencies to pairwise frequencies, DyPS considers the best of the SecureGenome [11] approach and pairwise-based LR-tests T_r [3] and/or Λ [12]. On singlewise frequencies, DyPS uses SecureGenome's strategy to remove SNPs in linkage disequilibrium, and very rare SNPs (i.e., SNP positions with MAF ≤ 0.05). After removing those SNP positions, DyPS runs the singlewise-based LR-test to identify the safe SNP positions.

In addition, DyPS runs and verifies the power achieved in a pairwise-based test (which can use the T_r and/or the Λ metrics) in order to decide which pairs of SNP positions can also have their pairwise frequency safely released. Since at this point SNPs are in linkage equilibrium (i.e., they have no statistical correlation), it is sufficient to avoid publishing their pairwise allele frequencies, and the pairwise frequencies that involve one of the two SNPs (and a different one) might be published nonetheless.

The exhaustive verification for pairwise frequencies follow a similar scheme, and can be executed in parallel of the verifications for singlewise frequencies. It is important noticing that T_r and Λ provide a membership metric for safe releases of pairwise statistics. However, they do not apply any SNP pruning mechanism (i.e., removal of dependent and very rare SNPs). Therefore, DyPS extends the state-of-the-art by not only offering a mechanism to safely release both types of allele frequencies but also accomplishing it in a dynamic fashion.

B.2 Pseudocode

At this point, DyPS had already selected a safe batch of requests to use to update test statistics, as explained in Appendix A. DyPS then checks whether aggregate statistics can be updated using the selected batch of requests. This process is shown in the algorithm reported in Figure 19. DyPS first collects the safe SNP positions that can be released given the selected genomes and all combinations of genomes considering up to f adversary biocenters (line 5). For each selected SNP, DyPS then collects the previous releases where it has been previously updated, and checks whether they involved genomes that participate in the current batch of genomes (lines 6 to 14). DyPS then generates and loops over all combinations of releases that share the same SNPs (lines 15 to 23). For each possible combination, DyPS executes the SNP selection software over the resulting set of genomes. If the SNP position s in $safe\_SNPs$ is also found to be safe in $testSet$, it means that it can be updated, otherwise, it is ignored. In the end, DyPS has a list of safe SNP positions to be updated (line 24).

```
1: procedure DyPS for Agg. statistic releases(S, Rels) // selected set of genomes, list of releases so far
2:     Input: set of genomes from selected biocenters.
3:     Output: set of SNP positions for safe aggregate statistics releases.
4:     Uses: SNPSelection(S, B, f): returns safe SNP positions for a set of genomes S and combinations of (B f) genome sets; shareGenomes(rel): checks if a release shares genomes with release rel; AllCombinations(Rel_shared_SNPS): create all combinations of releases that shares SNPs with Rels_shared_SNPS.
5:     safe_SNPs := SNPSelection(S, B, f)
6:     for s in safe_SNPs do // for each selected safe SNP
7:         for rel in Rels do// for each release so far
8:             if (s == rel.s) then // SNP position s has been released in a previous release rel
9:                 if (S.shareGenomes(rel)) then // candidate release S also shares genomes with previous releases rel
10:                    Rel_shared_SNPs.add(rel)
11:                end if
12:            end if
13:        end for
14:    end for
15:    for combRel in AllCombinations(Rel_shared_SNPs) do
16:        testSet := combRel + S // merge participating genomes in both releases
17:        checkSafeSNPs := SNPSelection(testSet)
18:        if (s in checkSafeSNPs) then
19:            continue // this SNP can be released
20:        else
21:            safe_SNPs.del(s) // cannot find a safe release this SNP this round
22:        end if
23:    end for
24:    return safe_SNPs // set of safe SNPs for candidate release S
25: end procedure
26: computeAggStatistics(safe_SNPs) // compute and release aggregate statistic over the set of selected SNPs
```

Figure 19. DyPS pseudocode for aggregate statistic releases.