nature portfolio

https://doi.org/10.1038/s43588-021-00044-9

Supplementary information

Citizen-centered, auditable and privacy-preserving population genomics

In the format provided by the authors and unedited

Supplementary Materials for

Citizen-Centered, Auditable and Privacy-Preserving Population Genomics

Dennis Grishin,[1,4*†] Jean Louis Raisaro,[2*†] Juan Ramón Troncoso-Pastoriza[3†], Kamal Obbad[4], Kevin Quinn[4], Mickaël Misbach[3], Jared Gollhardt[4], Joao Sa[3], Jacques Fellay[2,5], George M. Church[1,4], Jean-Pierre Hubaux[3]

[1]Harvard Medical School, Boston, MA, USA

[2]Precision Medicine Unit, Lausanne University Hospital, Lausanne, Switzerland

[3] IC School, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland

[4] Nebula Genomics Inc., San Francisco, CA, USA

[5] School of Life Sciences, École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland

[*]Corresponding authors: dgrishin@g.harvard.edu and jean.raisaro@chuv.ch.

[†]Co-first authors.

This PDF includes:

We begin the supplementary material by introducing a high-level description of the privacy and security concepts used in the paper. We continue by describing in details the material and methods that support our system for secure, privacy-preserving and auditable data sharing. Finally, we report and describe additional performance results that, due to space constraints, are not included in the main manuscript.

1 Security and Privacy Background

We first introduce the main cryptographic concepts used in our platform, with a special focus on privacy and security. In particular, we further develop: (a) secure computing techniques that enable data processing while avoiding undesired leakages to the computing party or parties; (b) data integrity and trust decentralization techniques, that enable immutable storage of relevant logging and transaction information guaranteeing tamper-proof integrity with no single points of failure, and (c) secure data release techniques, that enable sharing cleartext data (such as the results of a secure process) while mitigating malicious attacks on those data (e.g., membership inference or reconstruction attacks).

1.1 Secure computing

Confidentiality and privacy in collaborative processes are traditionally handled by the use of cryptographic privacy-enhancing techniques, especially homomorphic encryption and secure multi-party computation. In particular, our system requires database searches, which are more efficiently handled by equality-preserving encryption. We now give a high-level overview of these three technologies.

1.1.1 Homomorphic Encryption

Homomorphic Encryption (HE) enables computing on encrypted data without having to decrypt it first, consequently being an excellent enabler for secure processing in untrustworthy environments. By virtue of a mathematical homomorphism between the plaintext space and the ciphertext space, a server without access to the secret decryption key can obtain the result of group (addition or multiplication) or ring (addition and multiplication) operations on the plaintext by running their homomorphic counterparts on the ciphertext. Depending on the available homomorphism, HE systems can be either additive, multiplicative or algebraic (both additive and multiplicative). In order to be considered secure, any homomorphic cryptosystem must feature *semantic security*, which means that it is not possible for somebody without access to the secret decryption key to distinguish between encryptions of the same or different plaintexts. This requires the cryptosystem to be *probabilistic*, i.e., each encryption is randomized so that two encryptions of the same plaintext result in different ciphertexts with very high probability.

Elliptic Curve ElGamal

Our system requires an additively and probabilistic homomorphic cryptosystem to enable counts and histogram generation with encrypted data; for this purpose, we could have chosen either a lattice-based SHE, or an additively homomorphic cryptosystem. In order to (optionally) enable verifiability of the performed operations, we chose the latter, and more specifically, Elliptic Curve ElGamal (*1*), which can support efficient use of zero-knowledge proofs for correctness, contrarily to lattice-based SHE. Zero-knowledge proofs (*2*) are cryptographic constructions by which a prover can convince a verifier that a statement is true without revealing any further information.

It must be noted, though, that our system functionality is not bound to this choice and can be achieved with other cryptosystems. The encryption of a message $m \in \mathbb{Z}_p$ (the set of integers

between 0 and a big prime p), is a pair of points in the elliptic curve: $E_K(m) = (rG, mG+rK)$, where r is a uniformly-random nonce in $\mathbb{Z}_p$ (therefore the probabilistic nature of the encryption), G is a base point on an elliptic curve γ, and K is a public key.

The additive homomorphic property states that $E(\alpha m_1 + \beta m_2) = \alpha E(m_1) + \beta E(m_2)$ for any messages m_1 and m_2 and for any scalars α and β. In order to decrypt ciphertext $(rG, mG+rK)$, the holder of the corresponding private key k (such that $K = kG$) multiplies the first point rG and k yielding $k(rG) = rK$, and subtracts this point from the second term of the ciphertext $mG + rK$. The result mG is then mapped back to m, e.g., by using a hash table. We rely on fixed-point representation to encrypt floating values.

As described in the main text, we enable trust distribution by combining the public keys of several computing servers and building a collective public key. Decryption is then enabled as an interactive protocol, where each server sequentially performs partial decryption with its own secret key (which is still protected by the security properties of EC ElGamal, by virtue of the discrete logarithm problem).

1.1.2 Secure Multi-Party Computation

Secure multi-party computation (SMC) is an area of cryptography that aims at enabling several parties to evaluate a function on private data coming from distinct data sources without aggregating or sharing the input data (*3*). At the end of the protocol, the parties learn nothing more but the value of the function. SMC protocols are primarily based on either secret sharing (splitting the secret values into randomly generated *shares* that are distributed among all parties) or garbled circuits (hashing the truth tables of the to-be-computed circuit gates and executing them by obliviously transferring the inputs) (*4*). Recent progress on *oblivious transfer* made a lot of these protocols practical and even scalable (*5*), thus allowing for privacy-preserving machine learning computations. Various privacy-preserving supervised machine learning algo-

rithms have been proposed and analyzed in the SMC setting (*6*). There is an increasing interest in both training and prediction algorithms for machine learning under SMC (*7, 8*).

1.1.3 Equality-Preserving Encryption

The probabilistic nature of homomorphic cryptosystems, where the encryption function is a one-to-many function (it takes one same input to several different possible output values), prevents direct equality comparisons between ciphertexts. There are two options to enable this functionality: either (a) define the comparison function as a polynomial that can be homomorphically executed with the ring operations enabled by the cryptosystem; this solution is computationally complex, especially when the plaintext coefficients are not represented with a binary decomposition. Moreover, the result of the comparison would still be encrypted, requiring decryption or further homomorphic processing to be effectively used. (b) The other option is to use instead a non-probabilistic cipher in which the encryption of the same plaintext will always yield the same ciphertext; i.e., the encryption function is a one-to-one function (deterministic). This enables very efficient comparisons by just comparing the values of the ciphertexts.

In the case of EC ElGamal encryptions $E_K(m) = (rG, mG + rK)$, a deterministic version can be achieved by setting a fixed (secret) value for r, hence using only the second point of the encryption function ($E_{det,K}(m) = (mG' + rK)$), where G' is a base point of the curve, possibly different from G. In our distributed scenario, it is possible to split the value r and the base point G' among the computing nodes (analogously to the way the collective public key is split), and run an interactive protocol that converts a probabilistic encryption of a message into a deterministic one. Further details about this protocol can be found in (*9*).

It must be noted that, due to its equality-preserving property, deterministic encryption is susceptible to frequency attacks. An attacker that knows the frequency distribution of the clear-text terms can try to map them to the frequency distribution of the corresponding encryptions, hence

breaking it. In this work, we rely mainly on probabilistic homomorphic encryption, and whenever we need deterministic encryption, we apply additional countermeasures (see Generation of dummy data in the Methods Section) to avoid this kind of attacks.

1.2 Integrity, Accountability and Access Control

There are two main approaches to protect data integrity and enforce access control in an accountable way: centralized and distributed. We briefly introduce the advantages and disadvantages of each of them, justifying our choice of a distributed approach for our system.

1.2.1 Centralized Approach

Enforcing access control has been traditionally the job of centralized (cloud) services (*10*). These implement the logic that stores and interprets the rules applied to grant or deny access to the sensitive data, and also log the access requests and their outcomes. These services are typically manually configured and vulnerable to compromises. In personalized medicine, however, due to the sensitivity of the managed data and the inherently distributed nature of the data sources, relying on the existence and good will of such a provider or centralized authority poses significant risks that the multiple data holders are rarely willing to accept, especially when considering networks of clinical institutions or large sets of individuals. The single point of failure represented by the centralized authority becomes an obvious target for attackers, and represents a threat for the data in the whole network if compromised. Furthermore, the recent growth of the personal genomics industry has exacerbated existing privacy concerns due to widely publicized security breaches, non-transparent data sharing practices and government access to genomic data without consent.

1.2.2 Distributed Approach

In a distributed environment, a centralized authority is avoided, and accountability usually relies on trusting each and every of the stakeholders involved in the collaborative processes (federated network). Our approach prevents this weakness by relying on distributed ledger technologies (DLTs, a.k.a. blockchains) (*11–13*). DLTs are considered a core building block for many next-generation technologies relevant in multiple sectors of our future society, such as finance (*14*), healthcare (*15*) and e-democracy (*16*). The idea around distributed ledgers is to move from a trust-and-hope model, where a centralized authority provides service to clients, to a trust-but-verify setting where the abstraction of this centralized authority is implemented by a (sub)set of the participants, who replicate the actions of the authority and collectively agree (by majority) on the ground truth. This split of trust makes the subversion of the system a challenging task for an adversary, as he would need to stealthily compromise the majority of participants, instead of a single central authority, in order to change the *ground truth*.

1.2.3 Permissioned vs public blockchains

There are two types of blockchains that can be deployed in a distributed scenario: public and permissioned. The former are typically related to cryptocurrencies (e.g., Bitcoin), where there is no restriction to the entities that can join the management of the chain and modify its contents; the rules for updating the chain are normally based on *Proof of Work*, requiring that the participating entities (*miners*) perform some computationally costly operations before they are allowed to modify the chain by adding a new block. Besides not being environmentally friendly, these chains are susceptible to the so-called *51% attack*, in which a (minority) group of miners temporarily accumulate enough computational power to be able to drift the evolution of the chain at their will.

Contrarily, permissioned blockchains restrict the set of entities that have write-access to the

chain, therefore effectively limiting the blockchain maintainers to a set of authorized parties. The evolution of the chain is not based on *proof-of-work*, but on strict consensus rules that define the majority ratio needed to add new blocks to the chain. Therefore, permissioned blockchains are a perfect fit for distributing the trust among a set of authorized servers, out of which the users only need to trust a majority, hence effectively avoiding single points of failure.

1.3 Secure Data Release

Once data has been processed and is ready to be publicly released or shared with authorized entities, it has to be protected to avoid inference attacks that can single out individuals whose data has been used to produce the released results. Traditionally, statistical approaches are used to protect released data against these attacks, by means of modifications to the data that reduce its accuracy but limit the leakage that could lead to successful inferences. These approaches comprise de-identification and anonymization techniques for released individual data records, and differential privacy for released aggregated data.

1.3.1 Data De-Identification

A data record is de-identified when direct identifiers are removed (*17*). Following the Privacy Rule of the Health Insurance Portability and Accountability Act (HIPAA), de-identification process can be applied following the Safe Harbor rule (removal of 18 personal identifiers), or by expert determination. In either of both cases, after the removal of direct identifiers it may still be possible to re-identify the individual in the presence of unique combinations of indirect identifiers, such as genomic data; therefore, de-identified records cannot be considered anonymized.

1.3.2 Data anonymization

Data anonymization (*18*) is the result of processing personal data with the aim of irreversibly preventing re-identification of the data subject. Both direct and indirect identifiers are subjected to aggregation, generalization, suppression and/or randomization to mitigate re-identification risks. There are several metrics to determine the achieved degree of anonymization; one of the most popular is k-anonymity (*18*): A dataset fulfills k-anonymity whenever it contains at least k individuals with any chosen combination of quasi-identifiers. Therefore, each individual in the dataset is "hidden" within an anonymity set of at least k people.

1.3.3 Differential Privacy

A function applied over a dataset is called differentially private (*19*) whenever the (distribution of the) result of such function does not substantially change by the presence or absence of one individual in the dataset. Therefore, the release of a differentially-private result reveals negligible information about each of the individuals in the dataset. Differential privacy is usually achieved through randomization techniques; i.e., adding a controlled amount of random noise to the output of the to-be-computed deterministic "exact" function $f(.)$, such that the variance of the noise exceeds the maximum deviation of the function's output produced by the contribution of any individual as part of the input dataset. There are several notions of differential privacy; the one we use in this work is (ϵ, δ)-differential privacy (*19*). Formally, this notion is defined in the following way: Given two databases X and X' that differ in one record, a randomized mechanism κ is said to be (ϵ, δ)-differentially private if for all possible outputs y (query responses) of K, it holds that $Pr[\kappa(X) = y] \leq e^{\epsilon} \cdot Pr[\kappa(X') = y] + \delta$, where $Pr[.]$ represents the probability of an event. The informal interpretation of the previous expression is that the fact that two databases differ in one record does not significantly change the probability distribution of the output of the query. In practical terms, $\kappa(.)$ is usually defined as the deterministic (exact)

output of the desired query $f(.)$ plus noise with a bounded power, usually drawn from a Laplacian or Gaussian distribution. Recent work has demonstrated a query-answering mechanism that maximizes the utility of the system while achieving provable privacy guarantees (*20*). This approach can be applied to the differential privacy mechanism that is part of the system that we present here.

1.3.4 Protecting Genomic Data Release

While anonymization and de-identification techniques can, to some extent, mitigate the re-identification risk of clinical data, genomic databases have been proven vulnerable against several types of inference attacks. Membership inference has been popularized by Homer et al. back in 2008 (*21*). It consists of inferring whether some target individuals contributed their genomic data to a cohort while knowing only part of the target's data and summary statistics about the cohort. Another inference attack, variant inference (proposed by Humbert et al. (*22*)), can be carried out by relying on the intra-genome (between variants) and inter-genome (between relatives) correlations. Gymrek et al. have further shown that the identity of individuals could be inferred by leveraging genetic genealogy databases (containing surnames) with short tandem repeats on the Y chromosome (*23*). Shringarpure and Bustamante have also shown that it is possible to infer whether an individual is part of a group of interest by only asking binary questions about the presence or absence of alleles at different positions in the genome (*24*). It is worth noting that membership inference attacks have also been recently performed with relatively high success against genomic databases (e.g., Backes et al. (*25*)).

Consequently, genomic data cannot be effectively de-identified or anonymized, and it requires more strict protection mechanisms, such as encryption, to avoid re-identification risks.

2 Example of Secure Distributed Aggregate-Data Analysis: Privacy-Preserving GWAS Computation

The main goal of genetic research is to identify genes that are involved in human diseases and genome-wide association studies (GWAS) are one of the most popular methods used for this purpose. In particular, a GWAS checks if some specific genetic mutations occur more frequently in people with a particular disease than in people without the disease. The most common approach for GWAS is the case-control setup, which compares two large groups of individuals, one healthy "control" group and one "case" group affected by a disease. In particular, for each genetic mutation, the frequency of each mutation in the case group is compared to the frequency of the same letter in the control group. If the difference is statistically significant, then it means that the mutation in question is associated with the disease. The data of a genetic mutation on a set of cases and controls can be represented in a 2x2 contingency table. Several different statistical analysis methods can be applied to this table. For example, Pearson's chi-square (chi^2) test is one of the most conventional methods used to assess deviation from the null hypothesis that cases and controls have the same distribution of letter counts. As mentioned above, the privacy-preserving "aggregate-data analysis" functionality can be used in combination with the "cohort discovery" functionality to securely compute the input of 2-by-2 contingency tables for such chi-squared association tests without having to access the raw individual-level genomic data of the individuals. Let X be the set of genomic variants that a researcher wants to test for association with a given phenotype Y. To run a GWAS between X and Y, the researcher first runs a cohort discovery operation to identify the individuals in the system that match the inclusion criteria in the case and control groups. Second, the researcher runs an aggregate-data analysis operation on the two cohorts to obtain their corresponding size. Finally, by using the same two operations on the two identified cohorts, the researcher asks to identify for each genomic

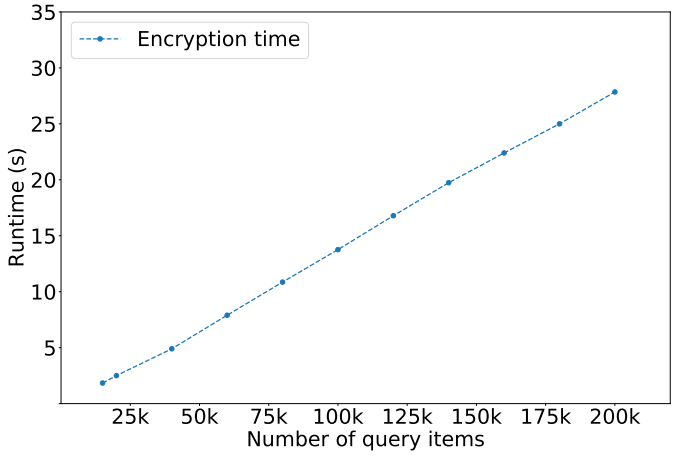
variant in X how many individuals carry a mutation and their corresponding count. Finally, by subtraction, the researcher can compute in the cleartext domain the number of individuals in the two cohorts that do not carry a mutation for each genomic variant, in order to build the complete set of 2-by-2 contingency tables and the chi^2 association test.

3 Supplementary Results

3.1 Performance Measurements

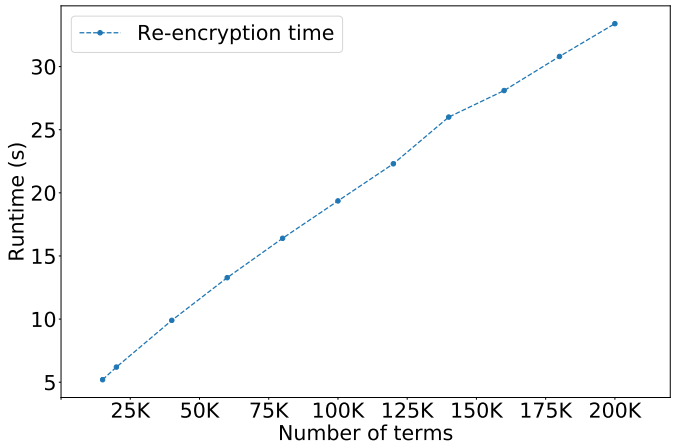
In addition to the performance evaluation reported in the main text, we run the following additional experiments with each measurement averaged over 10 independent runs:

- *Experiment 1 – Data preparation and loading time per data provider:* Supplementary Figure 1 shows the time needed for a data provider (individual) to encrypt an increasing number of clinical attributes and genetic variants and measured in a laptop with a Core i7 processor at 2.8 GHz and 16 GB of RAM.



Supplementary Figure 1: Runtime needed for encrypting an increasing number of clinical attributes and genetic variants at a data provider computer (before upload to the system).

Supplementary Figure 2 shows that the time needed by three computing nodes to re-encrypt the uploaded clinical attributes and genetic variants codes from homomorphic encryption to equality-preserving encryption is linear in the number of codes.

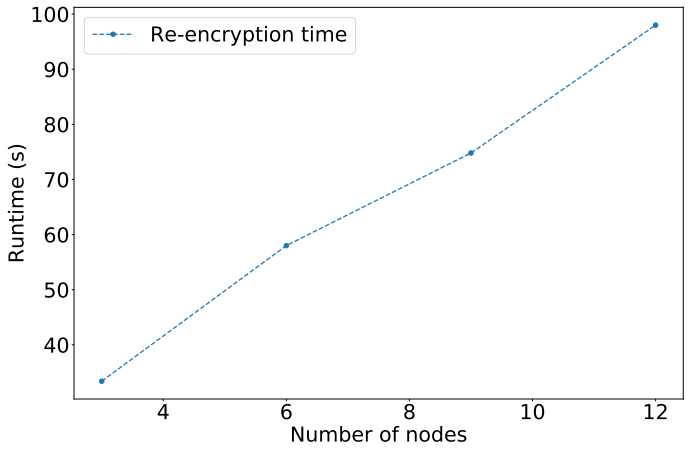


Supplementary Figure 2: System response (re-encryption) time for the upload of encrypted observations from a data provider for a growing number of variables, with three computing nodes.

Similarly, Supplementary Figure 3 shows the time needed to re-encrypt 200,000 uploaded clinical attributes and genetic variants codes from homomorphic encryption to equality-preserving re-encryption is also linear in the number of computing nodes involved in the protocol.

Finally, it is worth noting that the data upload only happens once per data provider and that the overhead introduced by the encryption and the secure re-encryption protocol is always (3 to 4 times) lower than the time it takes to do an insert operation of 1M variables in the database. This time indeed oscillates between 15 and 20 minutes regardless on the data being encrypted or not.

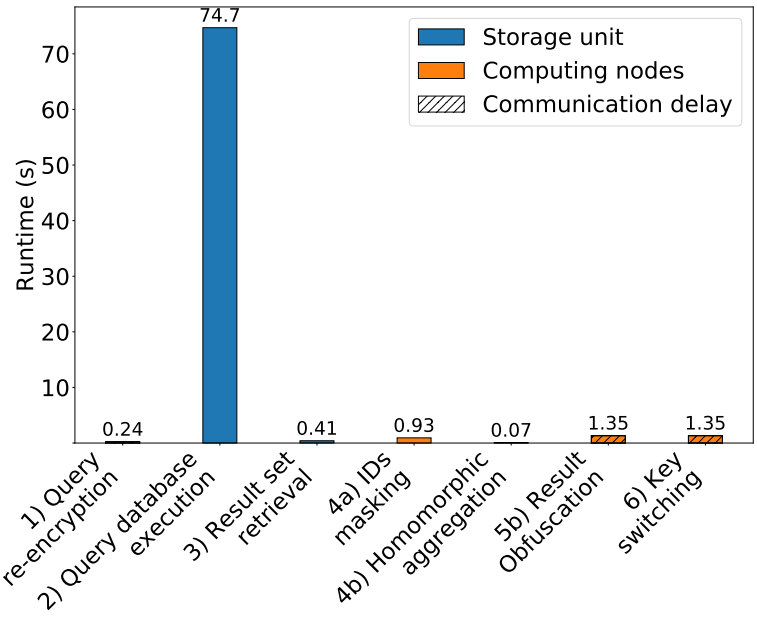
- *Experiment 2 – Query run time breakdown:* Supplementary Figure 4 explains the almost



Supplementary Figure 3: System response (re-encryption) time for the upload of encrypted observations from a data provider for a growing number of nodes, with a fixed set of 200k variables (individual variants).

negligible overhead introduced by the proposed encryption techniques. In particular, we observe that most of the query processing time is consumed by the execution of the query at the database level within the storage unit. This operation is not affected by the encryption, as it takes into account only the time needed to perform a set of standard SQL queries. This time mainly depends on the number of stored observations in the database and on the number of items composing the query.

- *Experiment 3 – GWAS computation:* Supplementary Figure 5 reports the response time of each of the subqueries sent for computing a GWAS as described in Supplementary Section 2. The total response time is linear with the size of the matching patient set, with a slope of approximately 21 ms per matching patient. The overhead of encryption grows linearly with the number of matching patients, but slower than the response time of the clear-text database. Therefore, the actual overhead is kept below 6% in practical scenarios. It must be noted though, that each of these subqueries can be run in parallel with

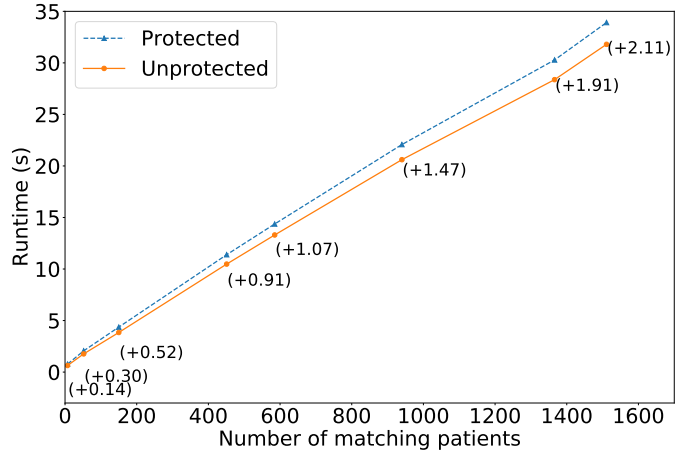


Supplementary Figure 4: Breakdown of query response time for 28 billion variables, 12 computing nodes and storage units, 10 query variables, and with a result set of 1511 individuals per node.

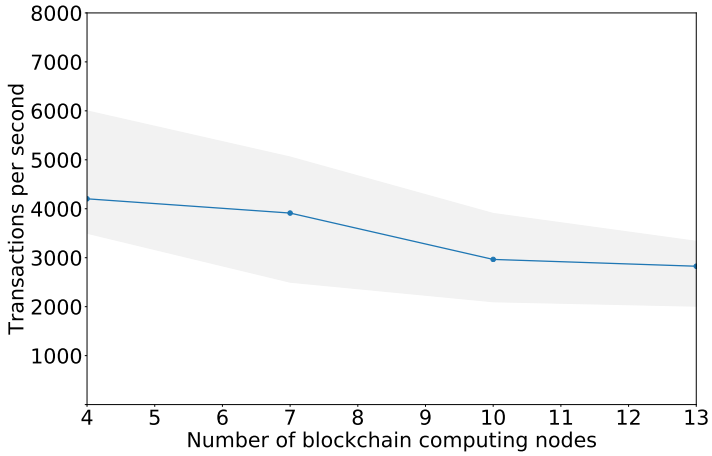
almost perfect scaling, both at the database engine level and at the cryptographic protocol level, which means that with the server setup that we are managing for the experiments, the response time is proportional to the number of variables divided by the number of available computing cores in each server.

- *Experiment 4 – Blockchain performance:* Supplementary Figure 6 reports the performance of our blockchain implementation, where the number of transactions per second is only slightly affected by the increase in the number of computing nodes. This demonstrates that the time to generate transactions for data and access policy uploads, for data discovery queries, and data access requests is negligible with respect to the time needed to execute these operations.

- *Experiment 5 – Storage overhead:* The storage overhead introduced by encryption affects



Supplementary Figure 5: Benchmark for computing a GWAS. Subquery response time (involving one single variable) for 28 billion variables, 3 computing nodes and storage units, and a varying number of patients. Subqueries can be easily parallelized.



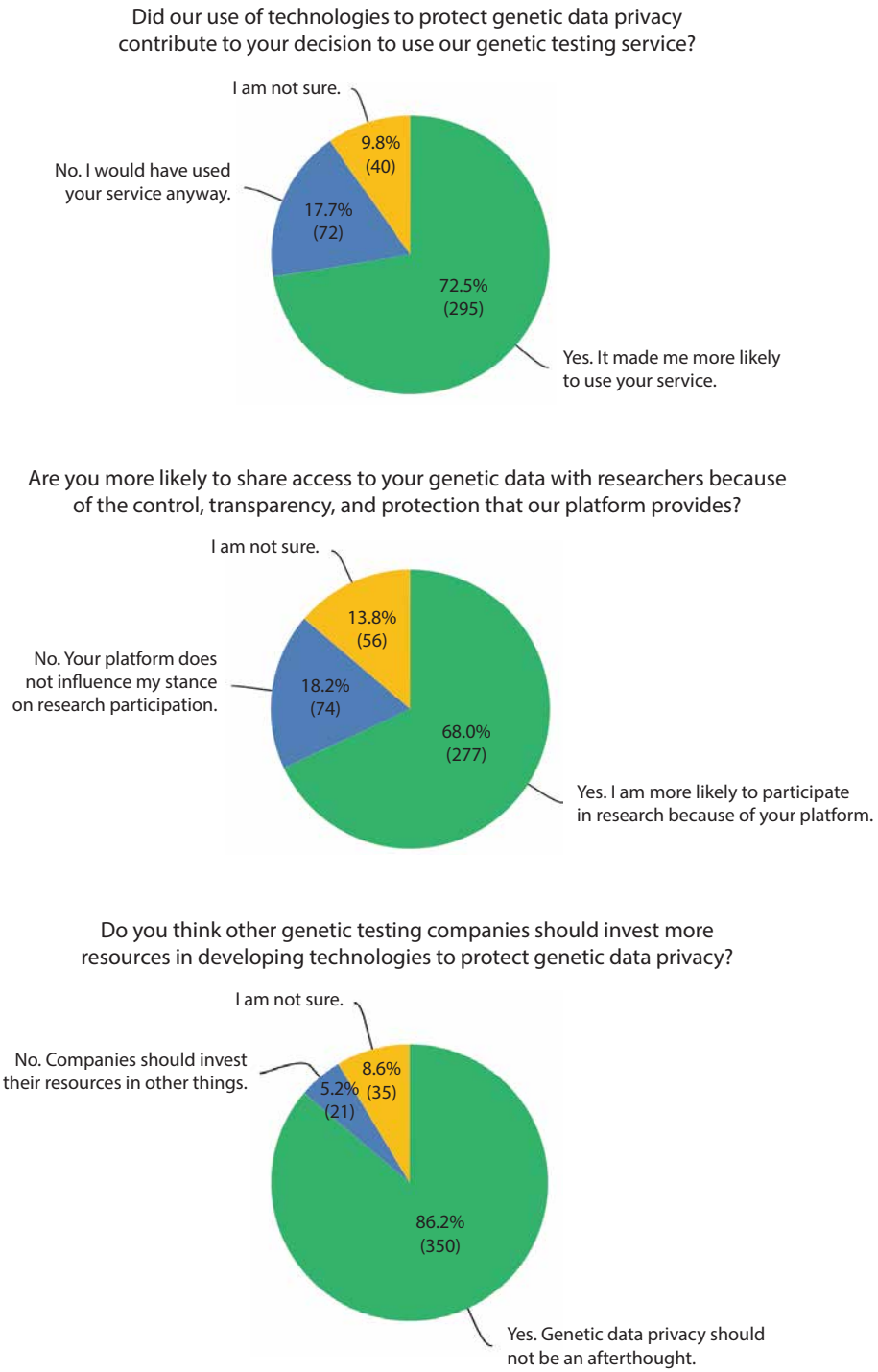
Supplementary Figure 6: Blockchain performance in terms of the number of transactions per second. The shaded area represents lower and upper throughput bounds.

only the set of unique observation codes used in the system, and it is in the order of 4x, as the use of equality-preserving converts each observation code, represented by a 64-bit integer, into a 32-bytes ciphertext. Depending on the specific distribution of codes across data providers, a varying number of dummy individuals is also added and must

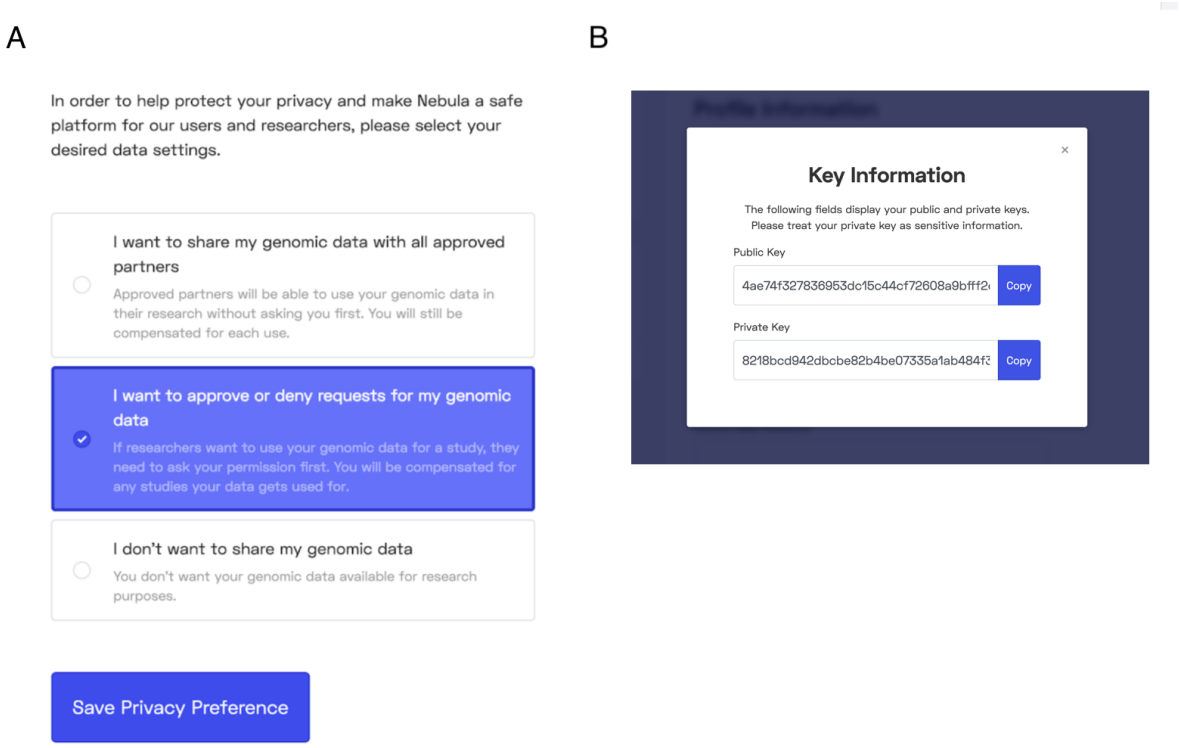
be considered as additional storage overhead. In the tested dataset we obtain an increase factor of 3.6x, which means that for each real data provider, 3.6 dummy individuals (on average) have to be generated.

3.2 User Feedback

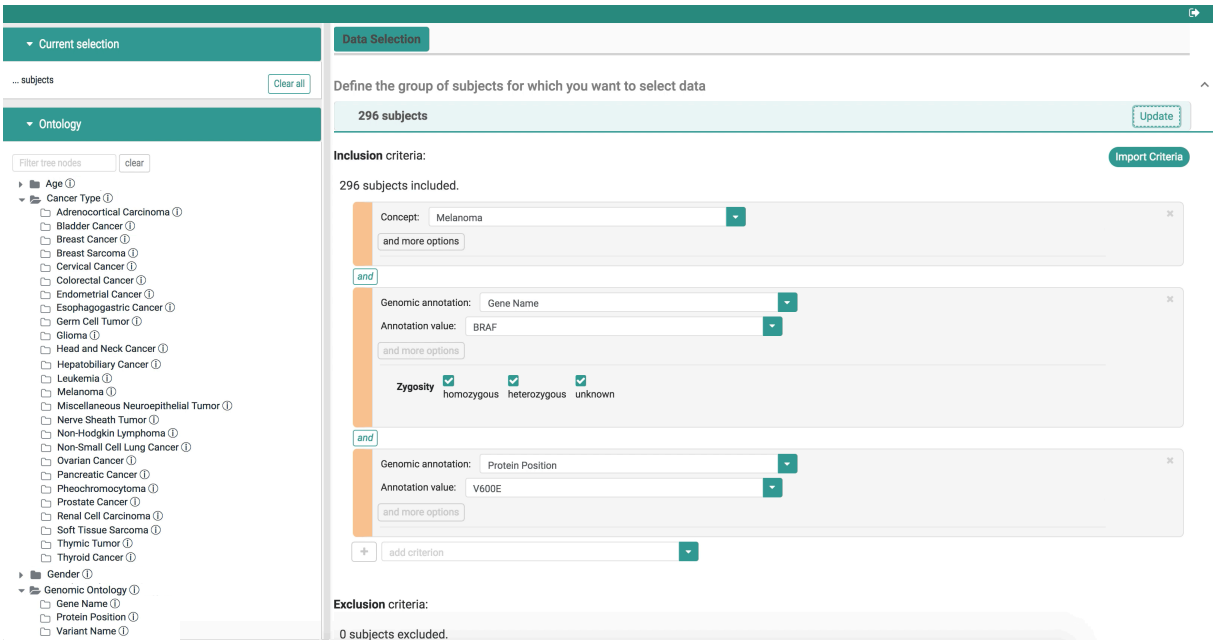
The blockchain component of our platform was deployed by Nebula Genomics, a genetic testing company. A survey of 407 Nebula Genomics users showed that the use of technologies to protect data privacy played a role in their decision to use a genetic testing service. The majority of participants were also more likely to share access to their genetic data with researchers due to the use of these technologies. Finally, the majority of survey participants believe that other genetic testing companies should prioritize data privacy protection (Supplementary Figure 7). A screenshot of the user interface that shows privacy setting and blockchain keys is shown in Supplementary Figure 8.



Supplementary Figure 7: A survey of individuals who used the blockchain component of our platform.



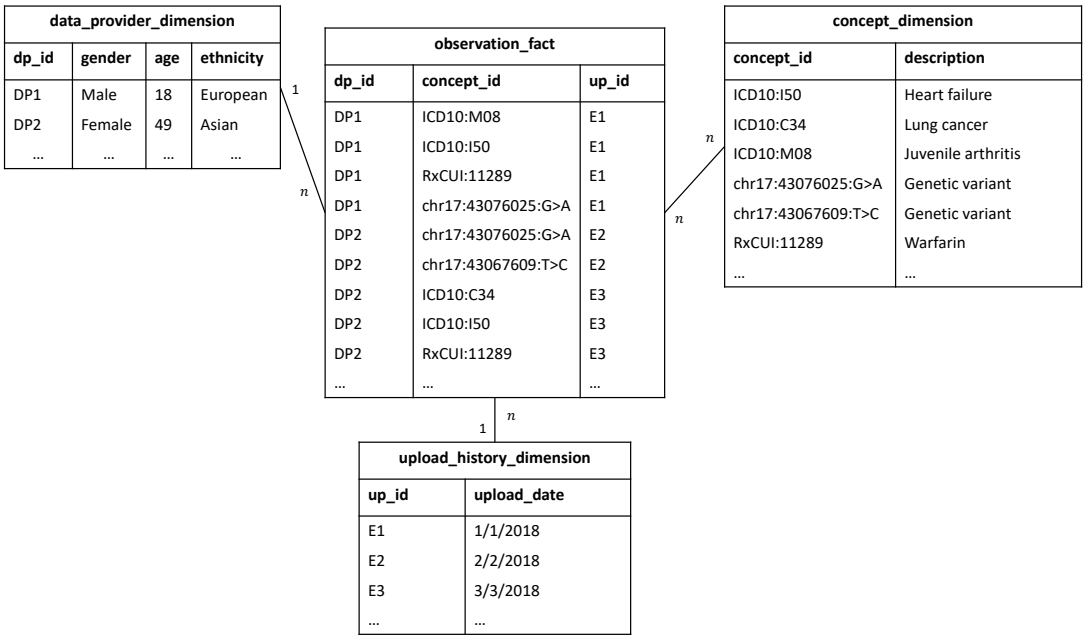
Supplementary Figure 8: Screenshot of the graphical user interface (for data providers) where (A) users select access policy and (B) can view their public and private keys.



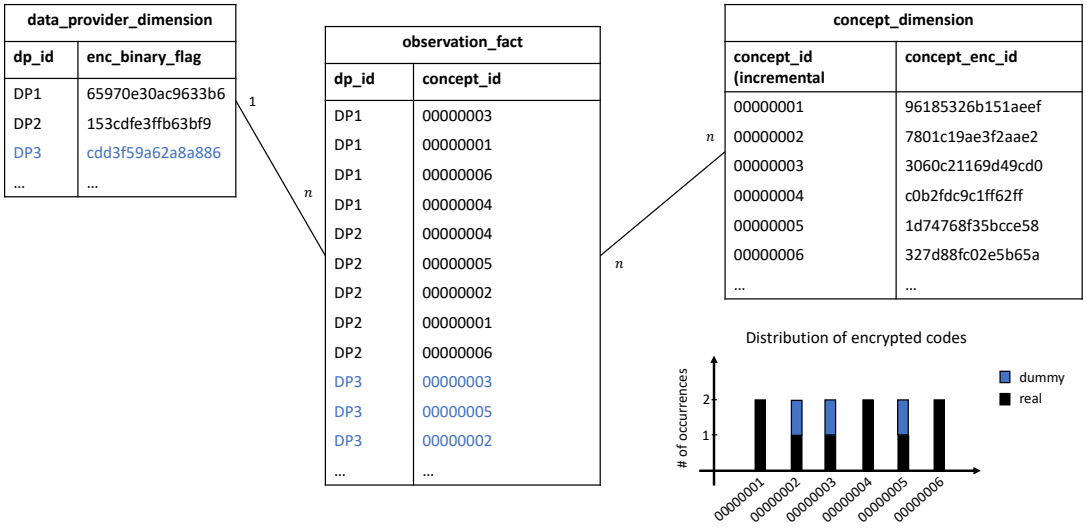
Supplementary Figure 9: Screenshot of the graphical user interface (for data queriers) where users can select query criteria and obtain aggregate-level statistics (individual count) on the identified cohort.

4 Supplementary Methods Figures

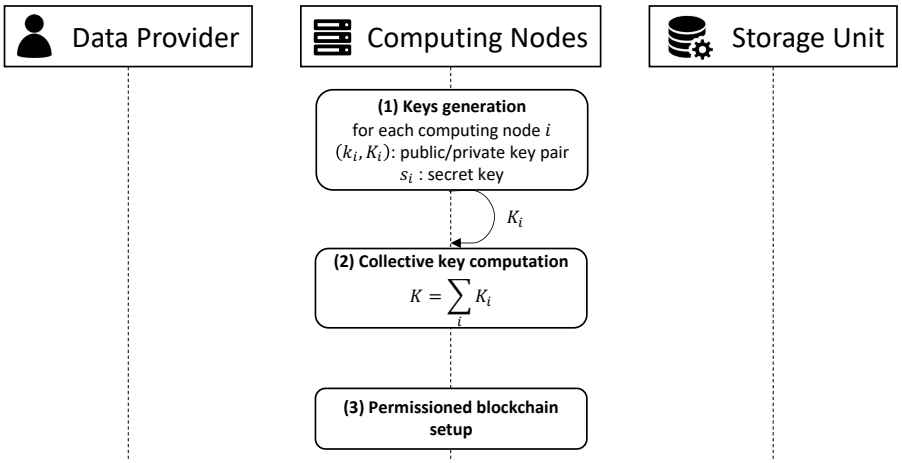
This section contains supplementary figures that are referenced in the Methods Section.



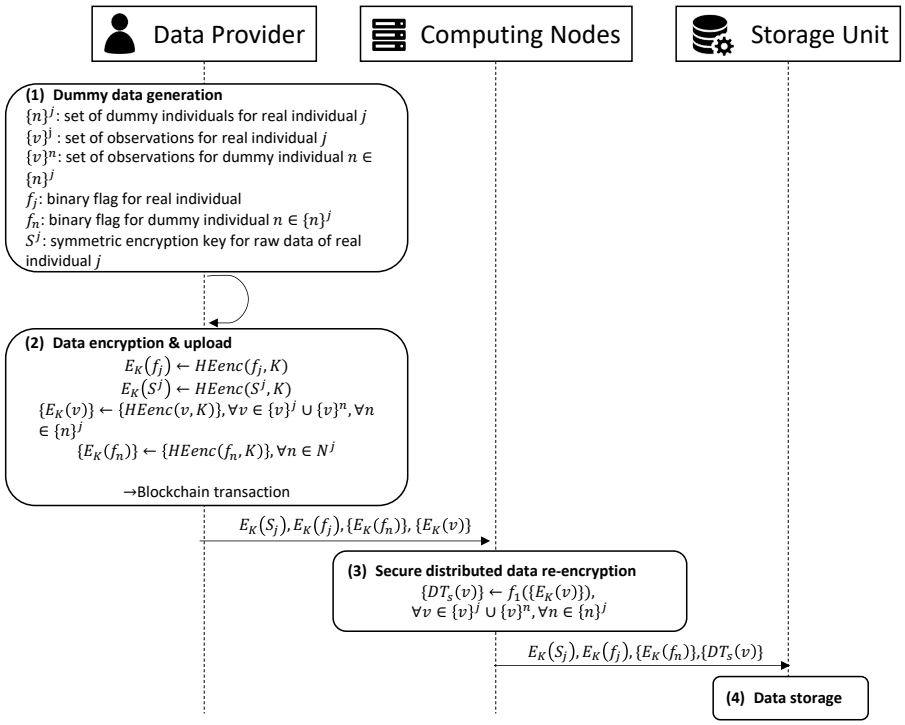
Supplementary Figure 10: Toy example of the "star-schema" data model storing data uploaded by two data providers DP1 and DP2. The *observation_fact* contains clinical and genetic observations for the two data providers (one observation per row). The meta-data about data providers is stored in the *data_provider_dimension* table. The *concept_dimension* table stores information about clinical and genetic variables (or concepts). The *upload_history_dimension* table stores meta-information about the data upload, e.g., the data upload date.



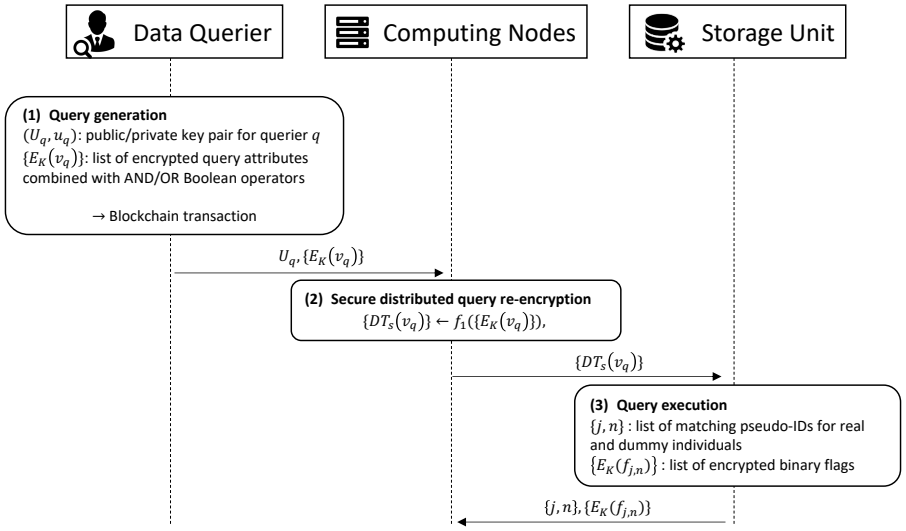
Supplementary Figure 11: Toy example of Supplementary Figure 10 with only the tables of the "star-schema" data model affected by data encryption. A dummy individual (DP3) with three observations has been added in order to flatten the distribution of encrypted observation codes.



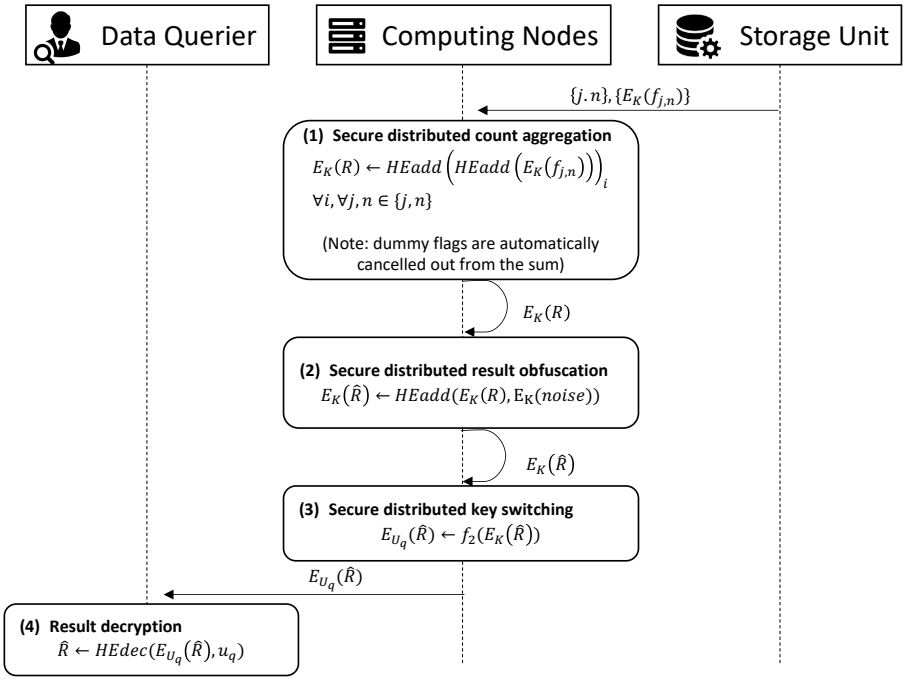
Supplementary Figure 12: System initialization phase sequence diagram.



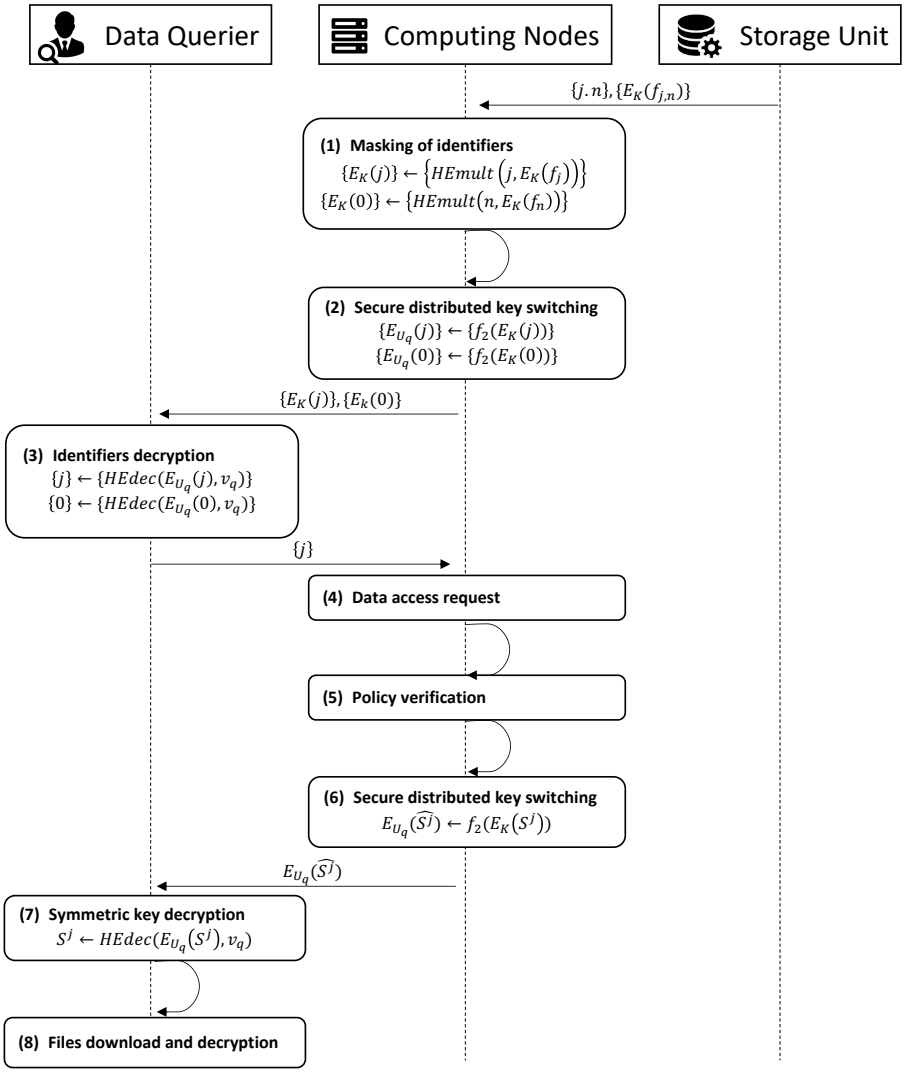
Supplementary Figure 13: Data preparation phase sequence diagram. For details on protocol f_1 see Supplementary Figure 17.



Supplementary Figure 14: Data discovery phase sequence diagram. For details on protocol f_1 see Supplementary Figure 17.



Supplementary Figure 15: Aggregate-Data Analysis sequence diagram. For details on protocol f_2 see Supplementary Figure 18.



Supplementary Figure 16: Raw-data access sequence diagram. For details on protocol f_2 see Supplementary Figure 18.

Secure distributed re-encryption protocol $f_1()$

$E_K(v) = (C_1, C_2) = (rG, v + rK)$ is the expanded representation of an EC-ElGamal encrypted observation v under the collective public key K, where r is a random nonce and G is the base point of the elliptic curve. This protocol consists of two rounds across all computing nodes CN_i.

Input: $E_K(v)$, s_i, k_i, K;
Output: $DT_s(v)$;

Round 1:

1: Each CN_i uses s_i to compute $s_iG + C_2$

2: $\rightarrow (\tilde{C}_{1,0}, \tilde{C}_{2,0}) = (rG, v + rK + \sum_i s_iG)$

Round 2:

3: Each CN_i computes $(\tilde{C}_{1,i}, \tilde{C}_{2,i}) = \left(s_i\tilde{C}_{1,i-1}, s_i(\tilde{C}_{2,i-1} - \tilde{C}_{1,i-1}k_i)\right)$

The equality-preserving version of the encrypted variable code is obtained by keeping only the second component of the resulting ciphertext.

4: $\rightarrow DT_s(v) = \tilde{C}_{2,n} = sv + \sum_i s_i sG$, where $s = \prod_i s_i$

Supplementary Figure 17: Secure distributed re-encryption protocol.

Secure distributed key switching protocol $f_2()$

$E_K(R) = (C_1, C_2) = (rG, R + rK)$ is the EC-ElGamal encryption of the query result with the collective public key K and U_q is the data querier's public key. This protocol consists of one round across all computing nodes CN_i.

Input: $E_K(R)$, k_i, K, U_q;
Output: $E_{U_q}(R)$;

Round 1:

1: $(\tilde{C}_{1,0}, \tilde{C}_{2,0}) = (0, C_2)$

2: Each CN_i generates a fresh random nonce v_i

3: Each CN_i computes $(\tilde{C}_{1,i}, \tilde{C}_{2,i}) = \left(\tilde{C}_{1,i-1} + v_iG, \tilde{C}_{2,i-1} - k_iC_1 + v_iU_q\right)$

4: $\rightarrow E_{U_q}(R) = (\tilde{C}_{1,n}, \tilde{C}_{2,n}) = (vG, R + vU_q)$, where $v = v_1 + \ldots + v_n$

Supplementary Figure 18: Secure distributed key-switching protocol.

References

1. Koblitz, N. Elliptic curve cryptosystems. *Mathematics of Computation* **48**, 203–209 (1987). URL https://www.ams.org/mcom/1987-48-177/S0025-5718-1987-0866109-5/.

2. Groth, J. & Sahai, A. Efficient non-interactive proof systems for bilinear groups. In Smart, N. (ed.) *Advances in Cryptology – EUROCRYPT 2008*, 415–432 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2008).

3. Cramer, R., Damgrd, I. B. & Nielsen, J. B. *Secure Multiparty Computation and Secret Sharing* (Cambridge University Press, New York, NY, USA, 2015), 1st edn.

4. Yao, A. C. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, SFCS '82, 160–164 (IEEE Computer Society, Washington, DC, USA, 1982). URL https://doi.org/10.1109/SFCS.1982.88.

5. Keller, M., Orsini, E. & Scholl, P. Mascot: Faster malicious arithmetic secure computation with oblivious transfer. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, 830–842 (ACM, New York, NY, USA, 2016). URL http://doi.acm.org/10.1145/2976749.2978357.

6. Nikolaenko, V. *et al.* Privacy-preserving ridge regression on hundreds of millions of records. In *2013 IEEE Symposium on Security and Privacy*, 334–348 (2013).

7. Mohassel, P. & Zhang, Y. Secureml: A system for scalable privacy-preserving machine learning. In *2017 IEEE Symposium on Security and Privacy (SP)*, 19–38 (2017).

8. Zheng, W., Popa, R. A., Gonzalez, J. E. & Stoica, I. Helen: Maliciously secure coopetitive learning for linear models. In *2019 IEEE Symposium on Security and Privacy (SP)* (2019).

9. Froelicher, D. *et al.* Unlynx: a decentralized system for privacy-conscious data sharing. *Proceedings on Privacy Enhancing Technologies* **2017**, 232–250 (2017).

10. Yu, S., Wang, C., Ren, K. & Lou, W. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *2010 Proceedings IEEE INFOCOM*, 1–9 (2010).

11. Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system (2008). Available online: http://bitcoin.org/bitcoin.pdf.

12. Wood, D. D. Ethereum: A secure decentralised generalised transaction ledger (2014).

13. Kogias, E. K. *et al.* Enhancing bitcoin security and performance with strong consistency via collective signing. In *25th USENIX Security Symposium (USENIX Security 16)*, 279–296 (USENIX Association, Austin, TX, 2016). URL https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/kogias.

14. Yermack, D. Corporate Governance and Blockchains. *Review of Finance* **21**, 7–31 (2017). URL https://doi.org/10.1093/rof/rfw074. http://oup.prod.sis.lan/rof/article-pdf/21/1/7/26322010/rfw074.pdf.

15. TECHNOLOGY, T. O. O. T. N. C. F. H. I. Use of blockchain in health it and health-related research challenge (2016).

16. Pilkington, M. *Blockchain Technology: Principles and Applications* (2015). Available at SSRN: https://ssrn.com/abstract=2662660.

17. Kayaalp, M. Modes of De-identification. *AMIA Annual Symposium Proceedings* **2017**, 1044–1050 (2018). URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5977668/.

18. Samarati, P. & Sweeney, L. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. Tech. Rep. (1998).

19. Dwork, C. Differential privacy. In Bugliesi, M., Preneel, B., Sassone, V. & Wegener, I. (eds.) *Automata, Languages and Programming*, 1–12 (Springer Berlin Heidelberg, Berlin, Heidelberg, 2006).

20. Cho, S. S. K. R. B. B., Hyunghoon. Privacy-preserving biomedical database queries with optimal privacy-utility trade-offs. *Cell Systems* (2020).

21. Homer, N. *et al.* Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays. *PLOS Genetics* **4**, 1–9 (2008). URL https://doi.org/10.1371/journal.pgen.1000167.

22. Humbert, M., Ayday, E., Hubaux, J.-P. & Telenti, A. Quantifying interdependent risks in genomic privacy. *ACM Trans. Priv. Secur.* **20**, 3:1–3:31 (2017). URL http://doi.acm.org/10.1145/3035538.

23. Gymrek, M., McGuire, A. L., Golan, D., Halperin, E. & Erlich, Y. Identifying personal genomes by surname inference. *Science* **339**, 321–324 (2013). URL https://science.sciencemag.org/content/339/6117/321. https://science.sciencemag.org/content/339/6117/321.full.pdf.

24. Shringarpure, S. & Bustamante, C. Privacy Risks from Genomic Data-Sharing Beacons. *American Journal of Human Genetics* **97**, 631–646 (2015). URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4667107/`.

25. Backes, M. *et al.* Identifying personal dna methylation profiles by genotype inference. In *2017 IEEE Symposium on Security and Privacy (SP)*, 957–976 (2017).