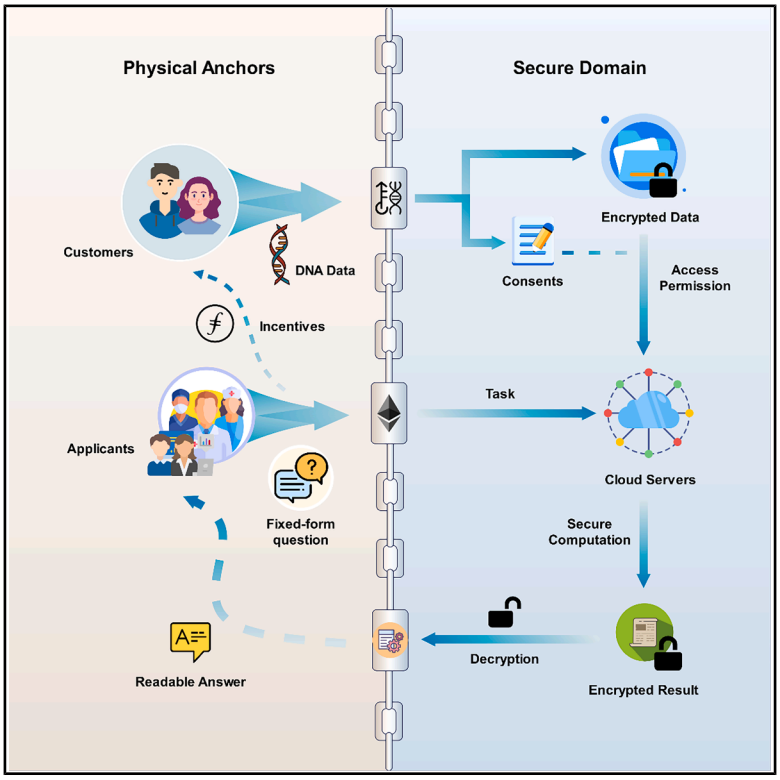
Cell Reports
Methods

Article

Toward owner governance in genomic data privacy with Governome

Graphical abstract



Highlights

- Governome is a framework to empower full control over personal genomic data
- Storage and sharing of genomic data in a secure domain without decryption
- Ensures transparent and verifiable data usage

Authors

Jingcheng Zhang, Yekai Zhou, Yingxuan Ren, ..., Yanmin Zhao, Junhao Su, Ruibang Luo

Correspondence

jhsu@cs.hku.hk (J.S.),
rbluo@cs.hku.hk (R.L.)

In brief

Zhang et al. present Governome, a framework that empowers individuals with comprehensive control over their genomic data. Governome ensures that data remain secure, private, and verifiable by leveraging modern cryptographic technologies. This system protects data integrity and allows for dynamic permission management, providing owners with full transparency over data usage.

Zhang et al., 2025, Cell Reports Methods *5*, 101171
September 15, 2025
https://doi.org/10.1016/j.crmeth.2025.101171

CellPress

Cell Reports Methods

CellPress
OPEN ACCESS

Article

Toward owner governance in genomic data privacy with Governome

Jingcheng Zhang,[1,3] Yekai Zhou,[1,3] Yingxuan Ren,[1] Man Ho Au,[2] Ka-Ho Chow,[1] Lei Chen,[1] Yanmin Zhao,[1] Junhao Su,[1,*] and Ruibang Luo[1,4,*]

[1]School of Computing and Data Science, The University of Hong Kong, Hong Kong 999077, China
[2]Department of Computing, The Hong Kong Polytechnic University, Hong Kong 999077, China
[3]These authors contributed equally
[4]Lead contact
*Correspondence: jhsu@cs.hku.hk (J.S.), rbluo@cs.hku.hk (R.L.)
https://doi.org/10.1016/j.crmeth.2025.101171

MOTIVATION In recent years, the proliferation of advanced sequencing technologies has made personal genomic data more accessible than ever before. This accessibility, while beneficial, raises significant concerns regarding data privacy, security, and ownership. Traditional data management systems often fall short of protecting individuals' genomic information, leading to potential misuse and privacy breaches. Out of such concerns, our research introduces Governome, a system that empowers individuals with comprehensive control over their genomic data. Governome ensures that data remain secure, private, and verifiable by leveraging blockchain technology, homomorphic encryption, and zero-knowledge proof. This system protects data integrity and allows for dynamic permission management, providing owners with full transparency over data usage.

SUMMARY

Advancements in sequencing technologies grant individuals unprecedented access to their genomic data. However, existing data management systems or protocols are inadequate in privacy protection, limiting individuals' control over their genomic information, hindering data sharing, and posing challenges for biomedical research. Therefore, demand exists for an owner-governed system fulfilling owner authority, life cycle data encryption, and verifiability simultaneously. Here, we realized Governome, an owner-governed data management system empowering individuals with real-time control over their genomic data. Governome leverages a blockchain to manage transactions and permissions, granting data owners dynamic permission management with full transparency on data usage. It uses homomorphic encryption and zero-knowledge proofs to enable genomic data storage and computation in an encrypted and verifiable form throughout its life cycle. Governome can support versatile genomic applications. We implemented and tested individual variant query, cohort study, genome-wide association study (GWAS) analysis, and forensics on 2,504 1000 Genomes Project (1kGP) genomes, demonstrating its robustness and scalability. Governome is open-source at https://github.com/HKU-BAL/Governome.

INTRODUCTION

The advent of affordable advanced sequencing technologies has empowered individuals to explore their health conditions through personal genomics.[1] However, with limitations in storage and analytical capabilities, many individuals opt for third-party services to host and analyze their genomic data. These services offer medical insights and analyses related to ancestry and disease susceptibility, expanding the utility of genomic data beyond clinical scenarios.

Despite the benefits, relying on third-party services introduces certain privacy risks. Individuals may wish to be aware of access relevant to their data but, in practice, are often presented with consent forms with limited options[2] that are often not thoroughly reviewed.[3] This leaves the data vulnerable to potential mishandling or misuse, as data sharing is often not well regulated. There are instances where commercial entities have shared genomic data with pharmaceutical firms in exchange for financial incentives, without informing data owners. Such a kind of data mismanagement can have immediate consequences. For example, individuals with high-risk genetic markers are denied life insurance coverage due to undisclosed genomic data usage.[4] Moreover, ensuring compliance with "the right to be forgotten" as outlined in the General Data Protection Regulation (GDPR),[5] which states the right to the

revocation of data access, can be a challenging task when utilizing third-party services. The revocation process typically involves users submitting requests to third parties, who are then required to comply with relevant regulations. The reliance on third-party compliance poses challenges in promptly executing data access revocations, let alone instant data access control.

The growing concerns surrounding genomic data privacy have spurred the development of various methods to protect data. For example, human genomic databases[6–8] host research-funded genomic data, which provide access only to authorized users to prevent potential mishandling or misuse. Another approach is to provide a unified application programming interface (API) for cross-institution genomic data sharing, thereby enabling federated data sharing with a security protocol. As an early attempt, Beacon Service from GA4GH[9] aims to achieve collaboration across databases through a distributed storage and sharing network. Since the potential reidentification through query analysis in these databases remains a critical privacy concern, practical solutions[10–13] have been proposed to mitigate risks while maintaining high data utility. Nevertheless, limitations[14] still exist in genomic data management systems that warrant technical improvements regarding better protection of genomic data privacy and security.

In addition to the efforts made in the existing databases, technology-based protections have emerged as practical solutions for genomic privacy. Recent review papers[15,16] emphasize the significance of genomic privacy from a social perspective and highlight privacy-enhancing technologies[17] as potential remedies, which involve applying mathematical, algorithmic, and hardware methods to genomic data. Early efforts focused on privacy-preserving data sharing and computation among institutions (also called data custodians). These methods are typically designed for specific genomics analysis tasks, such as cross-institutional single-gene disease diagnosis query,[18] genome-wide association study (GWAS),[19–21] and genetic imputation.[22–24] These task-specific protocols vary greatly in specializations and capabilities from different institutions. From a personal perspective, achieving comprehensive and timely control over one's personal genome still remains an ongoing challenge.

More recently, blockchain-based discussion[25] has offered a new insight into the field of privacy-enhancing technologies. Blockchain is a distributed ledger technology that enables multiple participants to engage in secure transactions and information sharing transparently without a central authority. It naturally aligns with the requirements of personal genome data owners to retain full authority over their genomic data without intermediaries as data custodians.[26] Therefore, starting in 2018, a well-known genomics security contest named iDASH[27] extended blockchain to one of its security computing tracks for the task of recording patients' data-sharing consents. There have been attempts[28] to store and share genomic data directly using a blockchain. While it ensures the security and immutability of transactions, the privacy of information stored on-chain is lost since any participant with read access to the system can directly access the raw genomic data. Another attempt introduced a citizen-centered method[29] that involved both secure computation and a blockchain-based system. However, it supports only simple genomics analysis tasks because only the addition operation is supported in its secure computation design, making it impractical for real-life genomics analysis tasks. It also lacks a measurement to avoid data owners and computing parties from providing false information, which is inevitable as the number of participants increases.

Challenges along the development of privacy-enhancing technologies have prompted us to reconsider their practical solution. We propose owner governance, which refers to three properties that should be met throughout the entire life cycle of genomic data: (1) the data owner retains full authority over her data, (2) the genomic data remain encrypted, and (3) the integrity of both the genomic data and the computation results is algorithmically guaranteed. Such a high degree of genomic data privacy protection can facilitate the contribution of personal genomic data to community benefits while preserving individual rights through real-time control over their genomic data. On the contrary, current solutions have evident limitations. In existing human genomic databases,[6–8] data access revocation is extremely difficult once the data have been shared, and queries about data usage logs and permission control are also entirely dependent on the credibility of the data custodian. Thus, establishing a comprehensive system for owner-governed genomic data management is imperative for addressing privacy concerns and empowering individuals in the genomic data landscape.

In this paper, we explored the pathways to achieving the three properties of owner-governance, namely owner authority (OA), life cycle data encryption (LDE), and verifiability (VER). We developed Governome, a realization of owner governance that fulfills all three properties. Governome utilizes a blockchain to manage all transactions and permissions, enabling data owners with dynamic permission management to be fully informed about all data usage. It uses homomorphic encryption (HE) and zero-knowledge proofs to enable genomic data storage and computation in an encrypted and verifiable form in its complete life cycle. In Governome, data owners are now able to share or revoke their genome in the system instantly. Querying entities can conduct analyses, including individual variant queries, cohort studies, GWAS analyses, and forensics. We benchmarked Governome for different applications and found that querying the population genotype distribution of a random SNP (single-nucleotide polymorphism) over 2,504 1000 Genomes Project (1kGP)[30] individuals can be efficiently completed in under 18 h on an ordinary server. Our experiments demonstrated that Governome can be applied to different genomic data management scenarios at scale. Governome is open source and available at https://github.com/HKU-BAL/Governome. To our best knowledge, Governome is the first realization of a secure, transparent, decentralized data management system that enables owner-governed genomic data management. We hope that Governome can set a new standard for genomic data privacy and data sharing in the personal genome era and, in turn, benefit personalized medicine and facilitate population genetics researchers on a larger scale.

RESULTS

Overview

In our conception, there are three properties that are required for the full fulfillment of owner governance in a genomic data

management system: OA, LDE, and VER. To this end, we developed Governome, which fulfills owner governance. Governome enables data owners to have real-time control of their genomic data with full transparency. No plaintext information is stored or generated in the system to eradicate any kind of data leakage. Data integrity and computation result authenticity are algorithmically ensured. Governome supports various genomic tasks, including variant query, cohort study, GWAS analysis, and forensics. We demonstrated Governome's performance with all variants of the 2,504 1kGP samples, suggesting its robustness when managing large-scale human genome projects and its potential to be scaled up to managing millions of samples.

Three properties of owner governance and their necessity

Here, we reason why a genomic data management system is capable of owner governance if, and only if, it simultaneously satisfies the following three properties.

(1) OA: owners have absolute and instantaneous control over their own genomic data. At any given time, data owners should be able to modify the access permissions of their genomic data in the system, including revoking data access entirely for any usage. OA also includes data owners' access to complete data usage logs that are guaranteed to be authentic.
(2) LDE: data must remain encrypted throughout its life cycle in the system, ensuring that they are never decrypted or accessed in raw form to protect data security. Encryption should be comprehensively applied to users' raw data or intermediate computation results in the stages of storage, exchange, and computation. No party should be able to directly access or reconstruct any raw information except for the query result provided by the system.
(3) VER: verifiability includes data integrity verifiability and computation process verifiability. Data integrity verifiability refers to the querying entities that initiate a query analysis in the system being able to verify whether the genomic data are free from tampering. Computation process verifiability requires the system to be able to provide evidence for the correctness of the results of any computing process.

OA is the core property of owner governance, which implies around-the-clock intermediary-free revocation and traceability. Intermediary-free revocation means that the data owners can break away from their previous commitments freely and at any time without any intermediary—they can be the ultimate decision-maker regarding data access of their own data. Traceability means data owners are fully informed, addressing information asymmetry challenges and enhancing control. The combination of decision-making and the right to be informed forms the basis of data owners' authority over their own data.

LDE is an inevitable requirement for ensuring data security in an owner-governed system. Unless proven otherwise, any disclosure of raw data, even to data owners, will result in potential risks such as information theft and storage device loss, which can have an irreversible impact on genomic data privacy. On the other hand, any party that acquires access to any raw data or intermediate results in plaintext means a deviation from OA since the party can maintain a copy with or without permission, which undermines data owners' right to decision-making.

VER ensures that the querying entities can always achieve the correct result, which is the foundation of usability. It prevents malicious participants from providing false information that could fake an identity or void research. The use of a blockchain implies crowdsourced data storage and computing. Hence, without a proper mechanism, a dishonest data provider or computing provider might act maliciously and cause permanent damage to the usability of the system. The principle of enabling VER is to trust no one and use mathematical and cryptographic tools to enforce data and computation integrity.

Without OA, data owners would effectively lose control over their genomic data. Without LDE, the genomic data within the system would face inevitable privacy risks when being used. Without VER, the system would lose its trustworthiness and, in the end, its usability. Therefore, as the next step of genomic data privacy, the three properties OA, LDE, and VER are integral.

How Governome realizes owner governance

We developed Governome, which fulfills the three properties simultaneously. To our best knowledge, it is the first realization of an owner-governance genomic data management system. As shown in Figure 1, Governome includes three layers: (1) a consensus layer to manage agreements among users; (2) a computing layer to manage the different forms of genomic data at various stages, including data storage, exchange, and analysis; and (3) an application layer as an interface for users to interact with the consensus layer and computing layer. The functionality of Governome is built upon the synergy of the three layers. Details about the techniques and design focuses at the three layers are shown in the feasible approaches to fulfill the three properties of owner-governance subsection of the STAR Methods.

The consensus layer is a blockchain that establishes the ownership of genomic data (see the techniques used at the consensus layer subsection of the STAR Methods). The blockchain stores (1) user permissions settings, (2) metadata and hashes for each query, and (3) the source code of supported genomic analysis tasks. Specifically, one's ownership of her genomic data should be universally recognized, and her modifications to the permissions of her genomic data should not have different versions across different nodes in the blockchain. For each query, (1) the encrypted result, (2) the individuals involved in serving the query, and (3) the metadata are stored on the blockchain. Owners can achieve auditability by either (1) checking requests that she replied to with the access token or (2) reconstructing the entire logs from the access requests. Moreover, with the support of metadata, hashes, and source code, the workflows in Governome are transparent and reproducible by anyone, thus resolving disputes. Importantly, Governome does not mandate decentralized storage of the genomic data itself; rather, it enforces decentralized storage of user metadata, keys, and access logs, which are typically only a few kilobytes in size when transferred. Adhering to the principle of LDE,

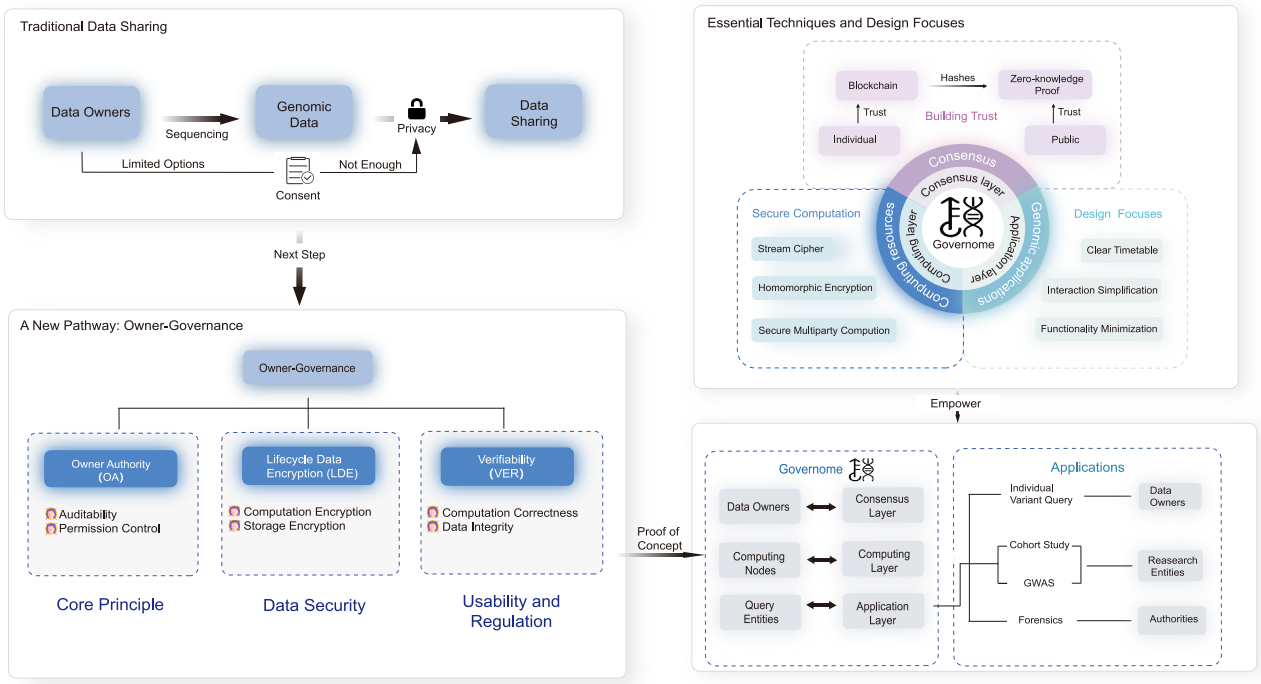


Figure 1. An overview of owner governance and its realization, Governome

Owner-governance requires three properties: (1) owner authority—owners must have absolute and instantaneous control over their owned genomic data; (2) life cycle data encryption—data must remain encrypted throughout its life cycle in the system; and (3) verifiability—includes data integrity verifiability and computation process verifiability. Governome, a realization of owner-governance, includes three layers that work synergistically, including (1) a consensus layer to manage user agreements, (2) a computing layer for secure computation, and (3) an application layer for genomic applications.

sensitive information is transmitted in encrypted form, ensuring both bandwidth efficiency and the security of network transmissions.

The computing layer is for aggregating the storage and computation resources of multiple parties with algorithms (see the techniques used at the computing layer subsection of the STAR Methods). The design of the computing layer focuses on (1) how genomic data are accessed, (2) how the genomic analysis tasks are performed, and (3) how multiple parties cooperate to participate in a task. The input of the computing layer is some encrypted data, while the output is fixed-form results of some genomic analysis tasks. Apart from the final output, all intermediate information is computable but cannot be decrypted. The computing layer is responsible for outputting reliable results for tasks, with the computing process being verifiable.

The application layer works as an interface for users who want to use the functions in Governome (see the design focuses at the application layer subsection of the STAR Methods). Considering the steep learning curve of cryptography and secure computation, a user-friendly interface is needed in Governome, while all modules related to privacy and security should be encapsulated within the consensus layer and computing layer. The design of the application layer, on the other hand, focuses on determining who can use Governome and how different users can utilize Governome, where users can simply ask questions predefined by the interface and receive responses. Moreover, as required by VER, when users question the reliability of computational results, they should be allowed to request evidence from the interface provided by the application layer and designate someone to verify the data integrity or computation integrity.

The workflow of Governome

The workflow of Governome is illustrated in Figure 2. To use Governome, a query entity can submit fixed-form questions to the interface defined by the research ethics committee. For example, one can ask, "What's the genotype distribution of rs6053810 for congenital heart disease patients?" After checking relevant on-chain consent, the consensus layer will send a request to the data owners for an access token (details are shown in the how to encrypt genomic data subsection of the STAR Methods), which can make part of their genomic data accessible to the computing layer. After the access tokens for all data owners involved have been collected, the computing layer will pull data from the storage nodes, perform secure computation, and return a readable answer to the query entity through the interface of the application layer (details are shown in the how computing layer works subsection of the STAR Methods). It is worth noting that both access tokens and genomic data are utilized in an encrypted form. Apart from the readable result, no other information is decrypted, thus fulfilling LDE.

The protocol sequence of each layer is illustrated in Figure 3. The workflow begins when a query entity poses a fixed-form question in the application layer. In the consensus layer, the

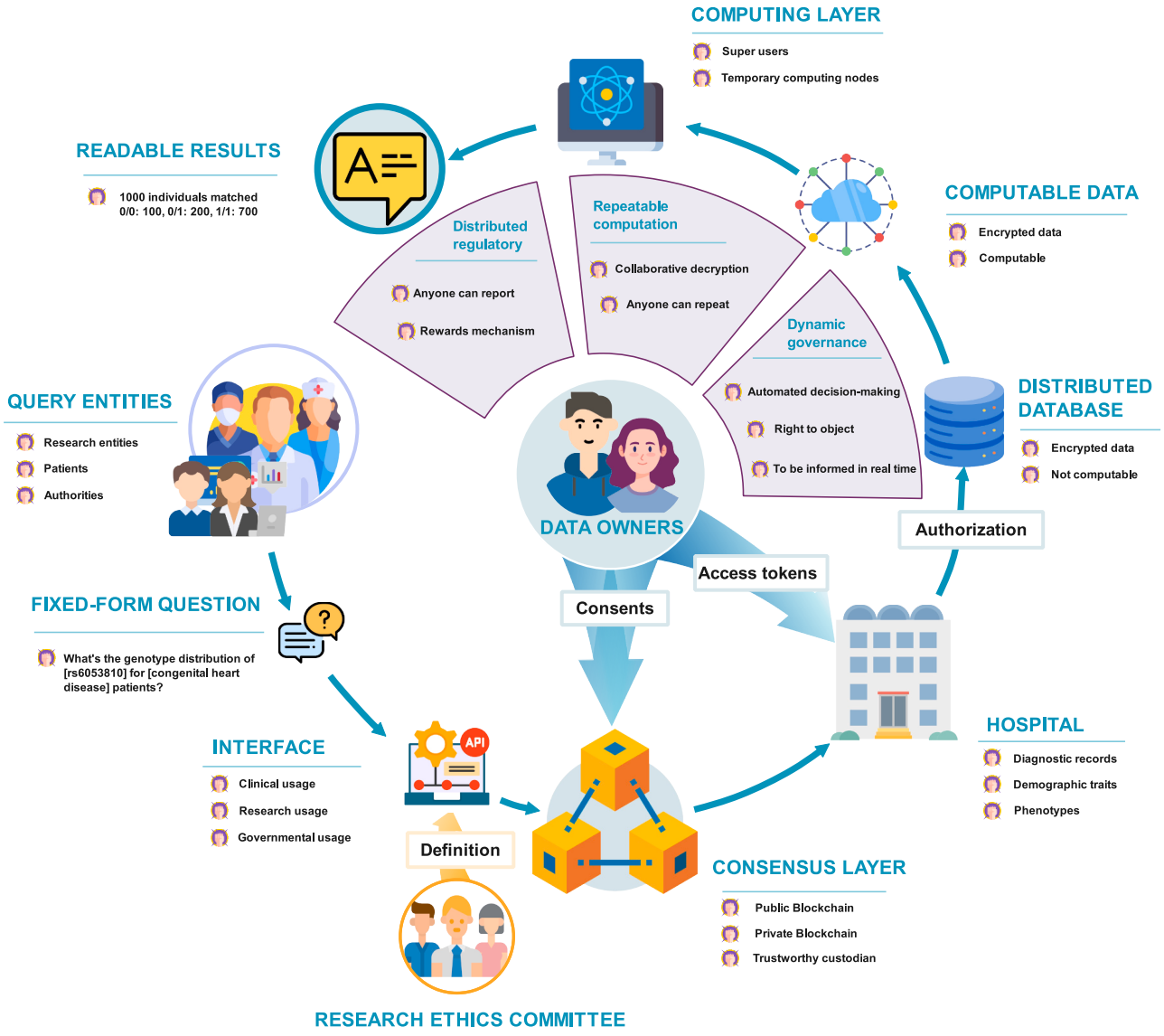


Figure 2. The workflow of Governome

A query entity can raise a fixed-form query through the interface of Governome. According to on-chain consents, the consensus layer will send requests to hospitals and subjects for access tokens that are indispensable for downstream homomorphic encryption-based computations. Next, the computing layer will pull relevant data for homomorphic encryption-based computations and return a readable result.

blockchain sends a request to the data owners and receives access tokens in response. These access tokens, along with the corresponding encrypted data, are then transmitted to computing nodes for processing the predetermined tasks. All computations occur within a secure domain, and the encrypted results are sent to the secure multi-party responsible for generating the computation metadata for decryption. Once the computing layer pipeline is complete, a readable result is returned to the query entity in the application layer.

In general, a data owner is required to be actively responding to requests (specifically, sending access tokens) from the blockchain; otherwise, her data cannot be accessed and would be excluded from analysis. However, it is impractical to require all data owners to be online and responsive around the clock. Therefore, in Governome, an option is given to data owners to register a precomputed access token so that Governome will skip the data owner and proceed with the token for computing. With this option, a data owner does not need to be active for her data to be used. The registered access token does not need to be recomputed until the next refresh of the computing layer. Details about the precomputed access token can be found in the precomputed access token subsection of the STAR Methods.

Supported genomic analysis tasks in Governome

The application layer has defined a list of genomic analysis tasks, including individual variant query, cohort study, GWAS analysis,

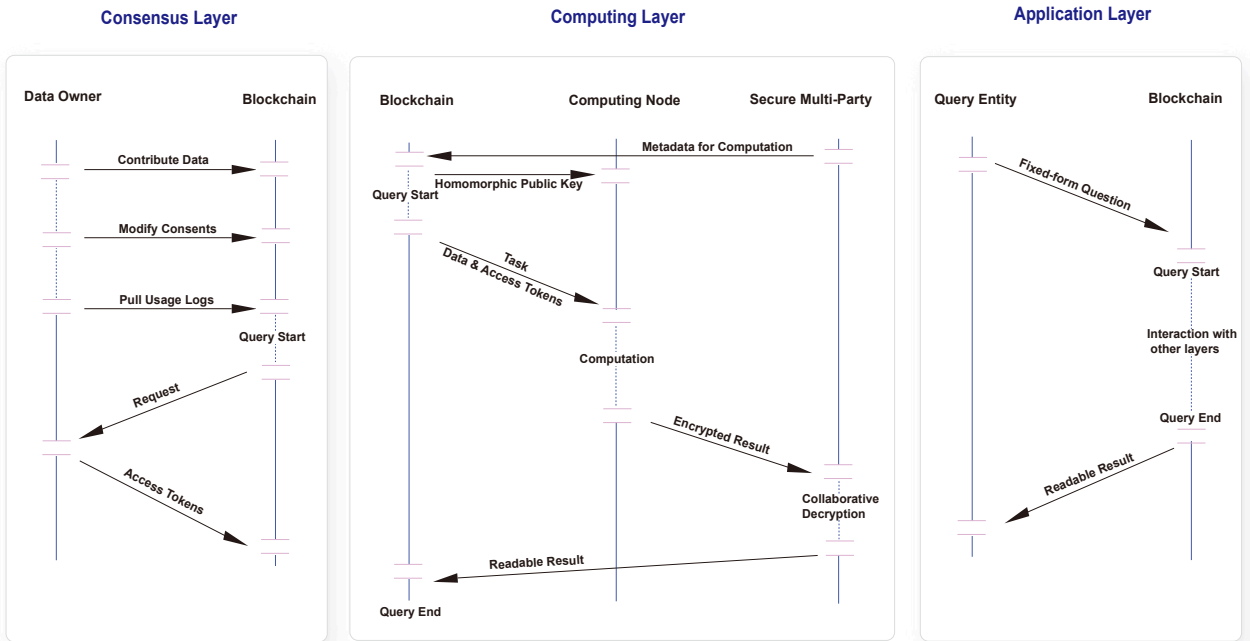


Figure 3. The protocol sequence of each layer

In the consensus layer, a data owner can modify consents or retrieve data usage logs after contributing their data. Upon request, the owner generates an access token for responses. In the computing layer, computing nodes process the predetermined tasks using the access tokens and data provided by the consensus layer. Once completed, the encrypted computation results are sent back to the secure multi-party that generated the metadata for decryption. Subsequently, the readable results are returned to the blockchain. In the application layer, the query entity interacts with the blockchain, allowing them to pose fixed-form questions and receive corresponding results.

and forensics. This section shows the functionalities of the genomic analysis tasks and who can use them.

An individual variant query allows data owners to explore their own genomic information. Interesting examples include the following: if someone is interested in whether she suffers from an alcohol flush reaction after consumption, she can check the variant rs671,[31] which causes aldehyde dehydrogenase 2 deficiency. If a male individual wants to know if he needs to prepare for early-onset hair loss, he can check the variant rs6152,[32] which increases the risk of baldness. In Governome, one can input an rsID[33] and get the result of her own genotype. Noteworthy, data owners can always keep their own copy of data for access without limitation. But if a data owner chooses to access their own data in Governome, for coherence, the data owner needs to prove he has access to his or her own data, like anyone else. In fact, one not having access to their own personal genome data for a certain period of time is possible. For example, a parental guide might be needed for access by children.

A cohort study allows users to examine the genotype distribution of interested rsIDs relevant to one or more demographics or phenotypes. GWAS analysis allows users to compare a disease cohort against a normal cohort at the interested rsIDs, with *p* values returned as results. For each cohort, Governome requires a minimum size of k individuals to be included for generating aggregated results from individual genomes, which is a common practice to reduce the risk of reidentification based on rare variants. That is, a cohort requires a minimum of k individuals to avoid the risk of being reidentified. The k in Governome is configurable, and Governome returns an error if an analysis forms a cohort with fewer than k individuals. Detailed descriptions of the algorithms used for GWAS can be found in the HE-based GWAS analysis subsection of the STAR Methods.

Forensic analysis fulfills public security and legal purposes, such as anti-human trafficking. Given a set of genotypes, Governome can return a list of matching individuals registered in the system. Such an application can bring high social value and is considered to be one of the most important applications of a huge-scale owner-governed genomics database, in addition to research and discovery. However, it is also dangerous, and it compromises personal identity if misused. Therefore, forensic analysis is exclusive to governmental authorities, and in Governome, we allow a data owner to exclude herself from all forensic analysis, observing our promise to give data owners ultimate control of their data. Forensic analysis can be conducted among all participating individuals in the system or in a smaller group shortlisted by hospitals according to some known demographic characteristics and phenotypes.

Based on the supported genomics analysis tasks available in Governome, we generally distinguished three types of users that demand different analysis permissions (Table 1). The three types are data owners, authorities, and research entities. The ability to perform an individual variant query is exclusively granted to data owners. Using a blockchain, data ownership is

Table 1. Genomics analysis tasks allowed for different user groups

	Individual variant query	Cohort study	Forensics
Data owners	✓	✓	×
Authorities	×	✓	✓
Research entities	×	✓	×

immediately confirmed, and an individual variant query is processed instantly. In contrast, forensic analysis is exclusive to authorities due to its risk of reidentification. All types of users are allowed to conduct cohort studies in Governome. For a cohort study, if there is a sample list with more than k individuals involved, the query is processed without further authentication or qualification reviews. The types of users are expandable, and the allowed tasks are configurable in Governome.

Noteworthy, although Governome has only implemented a few common genomics analysis tasks, it has no limit on having more tasks as long as they can be implemented at the application layer. However, any new tasks need to be sufficiently analyzed and discussed before being introduced into Governome to avoid unintended privacy risks.

Computational performance of Governome

We evaluated Governome's computational performance in (1) generating proofs for an access token and (2) HE-based computation. These are the two most computationally demanding procedures in Governome. Governome is implemented with the programming language Go v.1.21, and all benchmarks were done using the same programming language and version.

Generating proof for access token

As mentioned in the workflow of Governome, a data owner should respond to the blockchain and return an access token if consenting to the data access request. An access token is the encryption form of an 80-bit key[34] kept by a data owner. Apart from the access token, she needs to provide some evidence to show that her 80-bit key has not been tampered with. Here, we have chosen ZK-SNARK (zero-knowledge succinct non-interactive argument of knowledge)[35] as the solution to provide evidence. ZK-SNARK enables a data owner to prove, without revealing any part of the 80-bit key, that (1) she holds the valid 80-bit key according to a hash saved on-chain and (2) she submitted an access token generated from the valid 80-bit key. More details about why we have chosen ZK-SNARK are given in the techniques used at the computing layer subsection of the STAR Methods.

The time and memory consumptions are shown in Figure 4. We used a laptop with an Apple M1 CPU and 16 GB of RAM, mimicking an average setting of a data owner. The time consumption shows how long it takes to generate a proof, and it implies the minimum time a data owner can respond to a data request. The memory consumption shows the peak memory used to generate a proof, and it implies how much memory is needed in a data owner's device in order to respond to a data request. The 80 bits in a key can be used together to generate a proof or can be divided into smaller blocks to generate multiple proofs before merging into a single proof (details are given in the zero-knowledge proof for access token generation subsection of the STAR Methods). The memory consumption increases linearly with the block size, but the time consumption may vary.

Our benchmark showed that the memory consumption increased from 1.1 GB at block size 1 to over 16 GB at block size 40 or higher. The time consumption varied between block sizes and had an average of 57 s. Since a data owner's computational capacity is commonly limited to a cell phone or a laptop, low memory consumption is preferred. We have chosen block size 1 as the default of Governome, as it has minimal memory consumption with moderate time consumption. As a result, a data owner can generate all the necessary information to respond to a data request with a memory consumption of approximately 1 GB and a time consumption of about a minute, which are completely acceptable.

HE-based computation

HE[36] is a cryptographic technique that enables computation on encrypted data. In Governome, all data analyses are strictly HE-based computations conducted at the computing layer. Computing nodes at the computing layer are usually powerful servers with many CPU cores and much RAM. For the benchmarks in this section, we used a server with two 32-core Intel Xeon Platinum 8369C 2.9 GHz processors and 512 GB of RAM. For any genomic analysis tasks, the two computationally intensive steps are benchmarked, including (1) data conversion from stream ciphertext (smaller in size for storage but cannot be used for HE-based computing) to HE ciphertext (larger in size for HE-based computing) and (2) data analysis that uses HE-based computation. For samples, we used the 1kGP dataset[30] comprising the whole-genome variants of 2,504 individuals. The mock-up phenotypes of the 2,504 individuals were provided by the Hail library and are available from its tutorial.[37] The extracted phenotypes were already normalized as either binary or categorical variables. We divided the samples into five cohorts for our benchmarks according to the five superpopulations, Africans (AFR), admixed Americans (AMR), East Asians (EAS), Europeans (EUR), and South Asians (SAS), defined in 1kGP.

Individual variant query and cohort study. An individual variant query is the simplest task in Governome. Our benchmark showed that, using a single CPU core, querying any random variant in an individual used at most 15 min to return a result. A cohort study, in comparison, demands much more computation, especially when the cohort size is large and when GWAS analysis is needed. In a cohort study, Governome allows inputting rsIDs to specify the variant of interest and demographic characteristics and phenotypes for choosing samples. If a cohort study query generates no error, Governome will return the number of chosen samples and the genotype ratio of the chosen samples at each rsID. The performance of querying a variant in five cohorts and all 2,504 1kGP samples is shown in Figure 5. Generally, the time consumption of both data conversion and data analysis increased linearly with the number of samples in a cohort. Querying a variant in all 2,504 samples was finished in about 18 h (13 h and 16 min for data conversion and 4 h and 37 min for data analysis). More CPU cores can be used for parallel computing when querying more than one variant. The results show that Governome can support any population scale because the linear increase in computation matches the

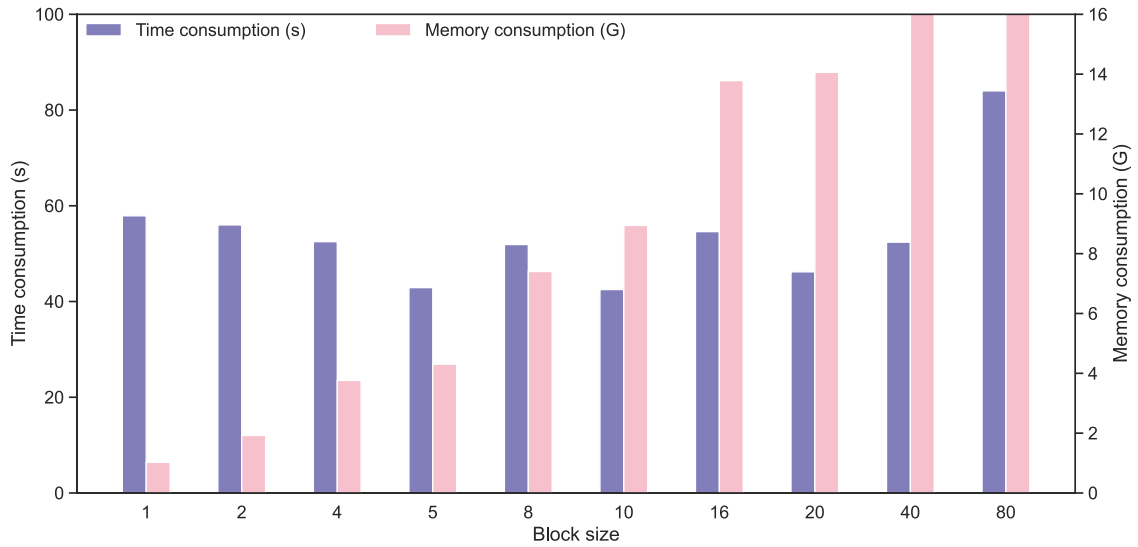


Figure 4. Performance of using ZK-SNARK to generate a proof for an 80-bit key using configurable block sizes ranging from 1 to 80

The memory consumption of block sizes 40 and 80 exceeded the available memory in our testing device (16 GB) and used memory swap. The exact numbers shown in the figure are given in Table S1.

expected linear increase in computing nodes when more individuals are introduced to the system.

For GWAS analysis, a *p*-value is calculated between the case samples and control samples at each rsID. We have chosen the phenotype "CaffeineConsumption" to divide the samples in each cohort into case (CaffeineConsumption > 4) and control (CaffeineConsumption $\leq$ 4) samples. The performance of the GWAS analysis on rs6053810 is shown in Figure 5. The results have shown that, while the number of samples of each cohort remained the same, data conversion took a similar amount of time, while data analysis took longer due to the algorithmically more complicated *p* value computation. Governome allows multiple data analysis tasks to be combined, so data conversion needs to be done just once.

Forensics. Forensic genetics relies heavily on analyzing short tandem repeat (STR) loci.[38] In our benchmark, we have chosen 13 STR loci[39] commonly used in forensics for analysis. One can carry out forensic analysis in Governome using cohort analysis at the interested STR loci with all samples in the system included in the cohort. However, the analysis will take excessively long, if it is not impossible to finish, when millions of samples are stored in the system. Therefore, we have added an auxiliary data block that stores only the genotype of the 13 STR loci for each individual (see the auxiliary data block subsection of the STAR Methods). The auxiliary data block is small and specific for forensic analysis. Thus, data conversion can be massively sped up when only forensic analysis is needed. Noteworthy, the auxiliary data block can include any number of variants for a specific analysis task in Governome, not limited to forensics.

When conducting a forensic analysis, an authority needs to input a list of STR loci with the genotype it is searching for. Additionally, demographic characteristics and phenotypes can be used to reduce the number of samples to be inspected. For each sample, Governome will output a Boolean vector showing a match or mismatch of genotype at each STR locus. As explained in the necessary supporting parties in Governome subsection of the STAR Methods, sample IDs in Governome are de-identified. Thus, outside Governome, in order to know the real identity of a matching individual, an authority needs to undergo legal procedures to get a warrant and further work with hospitals.

We tested the performance of the above design on 2,504 1kGP samples. Data conversion took 5 min and 51 s, while data analysis took 4 min. The performance is acceptable if the number of candidates for forensic analysis can be effectively narrowed down by known demographic characteristics and phenotypes.

Comparing Governome to previous solutions

In this section, we compare Governome against existing genomic data management systems on the three properties of owner governance. We evaluated OA on two dimensions called permission control and auditability. LDE was evaluated based on storage encryption and computation encryption, and VER was evaluated on data integrity verifiability and computation process verifiability. For each system being compared, we assigned either "fulfilled," "partially fulfilled," or "not fulfilled" for each of the six dimensions; the results are shown in Figure 6.

Existing human genomic databases are primarily government-funded centralized databases, such as dbGaP,[6] UK Biobank,[7] and AllofUS,[8] which typically directly provide the data to successful applicants. Some other databases are distributed, but they are still centrally managed and are more like an aggregation of government-funded centralized databases, such as GA4GH's Beacon.[14] These databases are centralized and have no capacity for owner governance, but these are among the most important human genome databases that have promoted the development of genomics in the past decade.

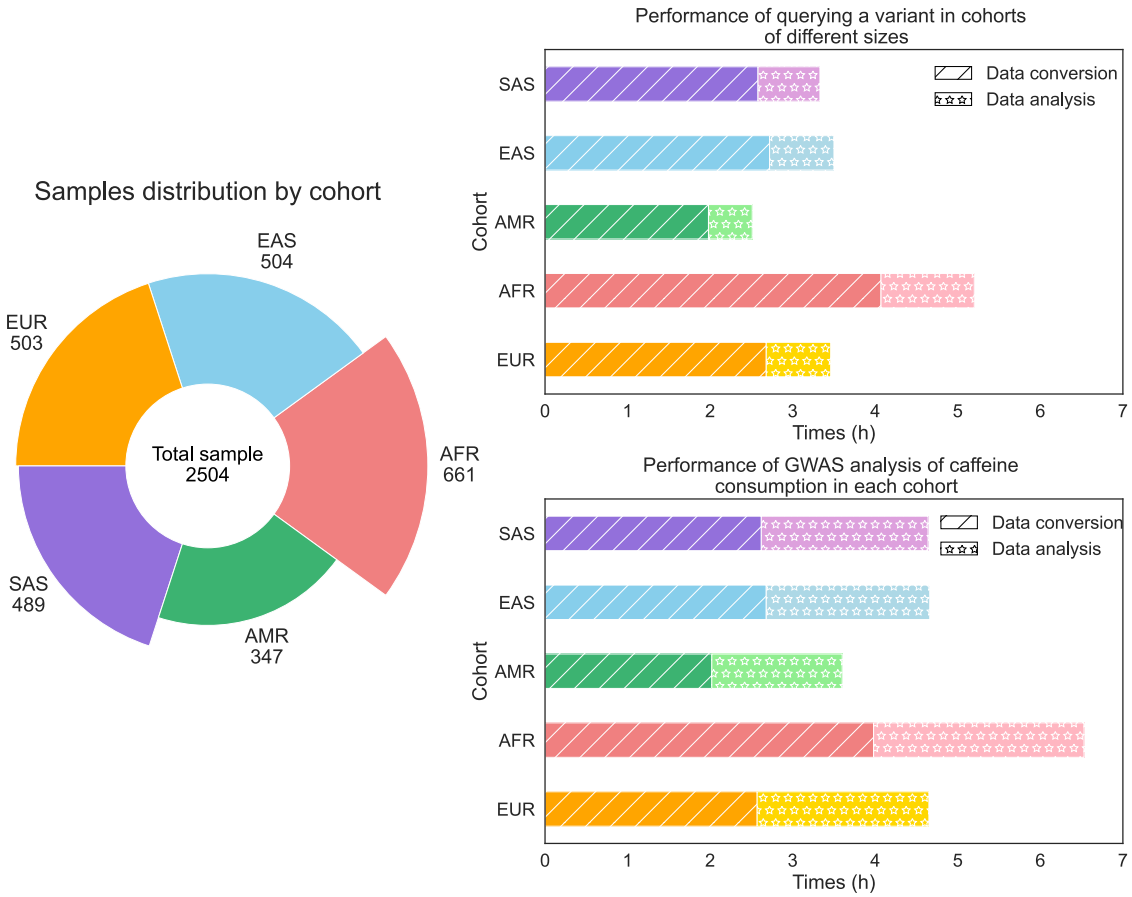


Figure 5. Performance of cohort study

This figure includes (1) querying a variant and (2) GWAS analysis of CaffeineConsumption. The exact numbers shown in the figure are given in Tables S2 and S3.

Unlike traditional human genome databases, Governome operates on a decentralized blockchain, enabling data owners to modify their consents instantly and access logs that are guaranteed immutable by the nature of blockchain, fulfilling the requirements of OA. All data within Governome are encrypted for their entire life cycle (no matter whether during computation or at rest), thereby satisfying the requirements of LDE. To fulfill VER, when being requested, a data owner needs to provide evidence for each operation she has conducted in the system through the zero-knowledge proof mechanism, so one cannot make up a piece of data in an analysis result without being further discovered. On each data request from the storage nodes, the computing nodes will perform a hash-based data integrity check against the metadata stored immutably on-chain. The zero-knowledge proof and hash-based validation used in Governome guarantee data integrity verifiability. For computation process verifiability, all homomorphic operations are repeatedly conducted by multiple computing nodes, making fraud in the computation process extremely hard if not impossible. Therefore, Governome is a realization that fully fulfills the three properties of owner governance. Other existing works are usually short of one or more of these three properties.

Grishin et al.[29] used a permissioned blockchain that restricts the set of entities that have write access to the chain, and hence, they have achieved full permission control and auditability. They used HE in both computation and storage. But while HE ciphertext is many times larger than the original text in size, they have chosen to only store the HE ciphertext of those who shared their data. This still allows an instantaneous revocation of data access, but resharing data requires uploading the HE ciphertext from the data owner again, which is a disincentive to data sharing and hinders active control of the data permission. The requirement that data owners always need to hold an unencrypted full copy of their data is also what we have avoided in Governome. Thus, we consider that Grishin et al. have only partially fulfilled storage encryption. In terms of computation encryption, we also consider Grishin et al.'s study as partially fulfilled because they used an HE scheme that supports only the addition operation, which limits the type and scale of genomic analysis tasks they can support. Grishin et al. have no mechanism to ensure that (1) data owners do not fake their uploads and (2) query entities can verify the correctness of the result obtained from the computing parties.

Gürsoy et al.[28] used a private blockchain to store BAM (sequencing raw data and alignments) and variant call format (VCF) data without encryption. They achieved auditability with the use of a blockchain, but since anyone can see everyone's

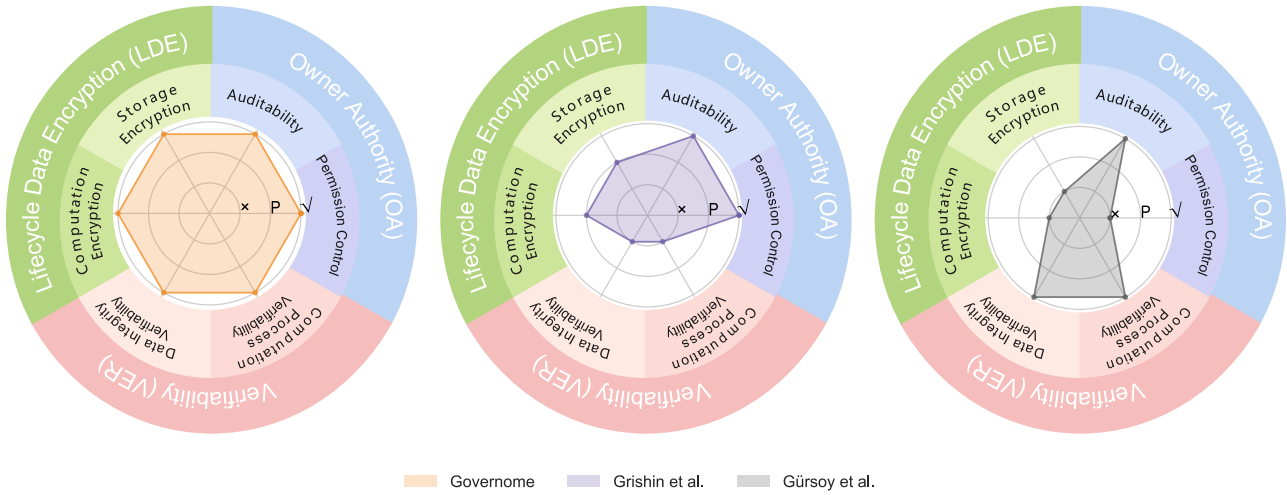


Figure 6. Comparing Governome against existing genomic data management systems

The evaluation metrics include permission control, auditability, storage encryption, computation encryption, data integrity verifiability, and computation process verifiability. Each metric is assessed with one of the following evaluations: fulfill (✓), partially fulfill (P), or not fulfill (×).

data on the chain, it puts any sort of attempt at permission control in vain. They obviously also lack computation encryption and storage encryption. However, they have fulfilled both data integrity verifiability and computation process verifiability because the genomic data are permanently stored on-chain, and any computation can be repeated and verified by others because the data are not encrypted.

DISCUSSION

In this paper, we reviewed the limitations of existing genomic data management systems. We reasoned that there were three properties that are essential to the full fulfillment of owner governance, a concept that we viewed as promising for the development of genomic data privacy. Based on our conception, we developed Governome, where data owners can exercise their personal rights outlined in GDPR, particularly the right to access, erase, and object. It is therefore our conviction that owner governance is the next crucial step to enhance data sharing between unfamiliar individuals and parties. In the most desirable outcome, research entities can conduct analysis without access approval, and data owners can not only receive financial incentives but also benefit from their genome throughout their lifespan.[1]

An often-overlooked aspect of genomic data management systems is the necessity for long-term financial investments to ensure user privacy, a requirement that significantly restricts the collection and sharing of centralized databases maintained by governments.[6–8] Actually, external incentives for genome sharing exist, with many companies and researchers willing to pay for the use and analysis of genetic data,[40] ultimately profiting from their products or research. Public blockchains present an ideal solution as a fair and reliable data-sharing mechanism due to their inherent immutability and traceability. They facilitate direct data sharing between data owners and query entities, eliminating the need for third-party data custodians that could introduce trust issues. Additionally, the financial attributes of blockchains enable the distribution of external incentives to data owners and computing nodes in the form of tokens, allowing them to benefit financially from their participation in research. This is why we advocate for the use of blockchain technology to establish a data-sharing platform that addresses the two most critical challenges in genomic data sharing: fairness and incentives.

At the computing layer, Governome uses Torus,[41] a third-generation HE technique, for HE-based computation. To our best knowledge, Governome is the first to use a third-generation HE technique for decentralized genomic data management and computing. The second-generation HE techniques used in previous solutions, such as TrustGWAS,[20] suffer from significant performance degradation when the computing becomes more complicated. This is because the efficiency of second-generation HE relies heavily on single-instruction multiple-data optimization,[42] which becomes difficult, if not impossible, when computation becomes complicated and contains excessive branches. Third-generation HE techniques have no such limitation and are more efficient, especially in data conversion. Such advancement has enabled Governome to support more complicated genomic data analysis tasks and future expansion.

There are several aspects that could be improved in Governome in future works. First, the current implementation supports only rsID as the variant index. rsID is reference genome agnostic and has been verified to be effective and sufficient for personal genomes at the stage performed by public personal genome sequencing services but is incapable of representing every single variant locus of a genome. While more and more personal genomes are now whole-genome sequenced, Governome should support the storage of VCF. In fact, Governome can be configured to use VCF to store variants easily. Secondly, Governome stores only genomic data and relies on hospitals or institutions that are eligible to host demographic and phenotypic data to shortlist qualifying samples for analysis. It is a practical design considering how most electronic health records are collected and organized. However, technically, Governome can also store and manage

demographic and phenotypic data. Moreover, genomic data in Governome are intended to be permanently stored. However, considering the significant advancements in quantum computing, Governome's security in the post-quantum era will become a new challenge. Currently, Governome is not quantum resistant. In the next step, we will explore optimizing cryptographic methods and privacy protocols to achieve post-quantum reliability.

Limitations of the study

Our study explores the potential future of decentralized management of personal genomes and presents a fully open-source prototype that demonstrates the feasibility of this approach. While this system showcases promise, it is inherently complex and requires significant expertise for effective operation and maintenance, particularly concerning latency and fault handling within blockchain networks. This complexity represents a limitation of our study, as the general applicability of our findings must still be validated through further research involving larger populations.

RESOURCE AVAILABILITY

Lead contact

Ruibang Luo (rbluo@cs.hku.hk) will fulfill any requests for additional information, data, and other resources.

Materials availability

This study did not generate new unique reagents.

Data and code availability

- All data supporting the findings, including source data and analysis results of this study, have been deposited at http://www.bio8.cs.hku.hk/governome/ and are publicly available as of the date of publication.
- All original code has been deposited at GitHub (https://github.com/HKU-BAL/Governome) and is publicly available as of the date of publication. The version used in this study has been released as a Zenodo archive: https://zenodo.org/records/16780081.
- Any additional information required to reanalyze the data reported in this paper is available from the lead contact upon request.

ACKNOWLEDGMENTS

R.L. was supported by Hong Kong Research Grants Council grants GRF (17113721) and TRS (T21-705/20-N), the Shenzhen Municipal Government General Program (JCYJ20210324134405015), and the URC fund from HKU.

AUTHOR CONTRIBUTIONS

Conceptualization, J.Z. and R.L.; methodology, J.Z., J.S., and R.L.; investigation, J.Z.; software, J.Z.; visualization, Y.R. and J.Z.; validation, Y.Z., L.C., and Y.Z.; supervision, R.L., J.S., M.H.A., and K.-H.C.; writing – original draft, J.Z., J.S., R.L., and Y.R.; writing – review & editing, all authors.

DECLARATION OF INTERESTS

The authors declare no competing interests.

STAR★METHODS

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

- KEY RESOURCES TABLE
- METHOD DETAILS
  - Feasible approaches to fulfill the three properties of owner-governance
  - Storage and computation setup in Governome
  - Necessary supporting parties in Governome
  - Data structure and cryptographic algorithms in Governome
- QUANTIFICATION AND STATISTICAL ANALYSIS
  - Zero-knowledge proof generation with gnark
  - HE-based computation with TFHE-go

SUPPLEMENTAL INFORMATION

Supplemental information can be found online at https://doi.org/10.1016/j.crmeth.2025.101171.

Received: November 26, 2024
Revised: May 20, 2025
Accepted: August 14, 2025
Published: September 9, 2025

REFERENCES

1. Rehm, H.L. (2017). Evolving health care through personal genomics. Nat. Rev. Genet. *18*, 259–267.
2. Niemiec, E., and Howard, H.C. (2016). Ethical issues in consumer genome sequencing: Use of consumers' samples and data. Appl. Transl. Genom. *8*, 23–30.
3. Geier, C., Adams, R.B., Mitchell, K.M., and Holtz, B.E. (2021). Informed consent for online research—is anybody reading?: assessing comprehension and individual differences in readings of digital consent forms. J. Empir. Res. Hum. Res. Ethics. *16*, 154–164.
4. Klitzman, R., Appelbaum, P.S., and Chung, W.K. (2014). Should life insurers have access to genetic test results? JAMA *312*, 1855–1856.
5. Voigt, P., and Von dem Bussche, A. (2017). The EU General Data Protection Regulation (gdpr). A Practical Guide, 1st Ed. (Springer International Publishing).
6. Tryka, K.A., Hao, L., Sturcke, A., Jin, Y., Wang, Z.Y., Ziyabari, L., Lee, M., Popova, N., Sharopova, N., Kimura, M., and Feolo, M. (2014). NCBI's Database of Genotypes and Phenotypes: dbGaP. Nucleic Acids Res. *42*, D975–D979.
7. Allen, N.E., Sudlow, C., Peakman, T., Collins, R., and biobank, U. (2014). UK Biobank Data: Come and Get it (American Association for the Advancement of Science).
8. The All of Us Research Program Investigators (2019). The "All of Us" research program. N. Engl. J. Med. Overseas. Ed. *381*, 668–676.
9. Global Alliance for Genomics and Health (2016). A federated ecosystem for sharing genomic, clinical data. Science *352*, 1278–1280.
10. Raisaro, J.L., Tramèr, F., Ji, Z., Bu, D., Zhao, Y., Carey, K., Lloyd, D., Sofia, H., Baker, D., Flicek, P., et al. (2017). Addressing Beacon re-identification attacks: quantification and mitigation of privacy risks. J. Am. Med. Inf. Assoc. *24*, 799–805.
11. Aziz, M.M.A., Ghasemi, R., Waliullah, M., and Mohammed, N. (2017). Aftermath of bustamante attack on genomic beacon service. BMC Med. Genomics *10*, 43–54.
12. Wan, Z., Vorobeychik, Y., Kantarcioglu, M., and Malin, B. (2017). Controlling the signal: Practical privacy protection of genomic data sharing through Beacon services. BMC Med. Genomics *10*, 39–100.
13. Venkatesaramani, R., Wan, Z., Malin, B.A., and Vorobeychik, Y. (2023). Defending against membership inference attacks on beacon services. ACM Trans. Priv. Secur. *26*, 1–32.
14. Shringarpure, S.S., and Bustamante, C.D. (2015). Privacy risks from genomic data-sharing beacons. Am. J. Hum. Genet. *97*, 631–646.
15. Bonomi, L., Huang, Y., and Ohno-Machado, L. (2020). Privacy challenges and research opportunities for genomic data sharing. Nat. Genet. *52*, 646–654.

16. Wan, Z., Hazel, J.W., Clayton, E.W., Vorobeychik, Y., Kantarcioglu, M., and Malin, B.A. (2022). Sociotechnical safeguards for genomic data privacy. Nat. Rev. Genet. *23*, 429–445.
17. Cho, H., Froelicher, D., Dokmai, N., Nandi, A., Sadhuka, S., Hong, M.M., and Berger, B. (2024). Privacy-Enhancing Technologies in Biomedical Data Science. Annu. Rev. Biomed. Data Sci. *7*, 317–343.
18. Jagadeesh, K.A., Wu, D.J., Birgmeier, J.A., Boneh, D., and Bejerano, G. (2017). Deriving genomic diagnoses without revealing patient genomes. Science *357*, 692–695.
19. Cho, H., Wu, D.J., and Berger, B. (2018). Secure genome-wide association analysis using multiparty computation. Nat. Biotechnol. *36*, 547–551.
20. Yang, M., Zhang, C., Wang, X., Liu, X., Li, S., Huang, J., Feng, Z., Sun, X., Chen, F., Yang, S., et al. (2022). TrustGWAS: A full-process workflow for encrypted GWAS using multi-key homomorphic encryption and pseudorandom number perturbation. Cell Syst. *13*, 752–767.e6.
21. Wang, X., Tang, H., Wang, S., Jiang, X., Wang, W., Bu, D., Wang, L., Jiang, Y., and Wang, C. (2018). iDASH Secure Genome Analysis Competition 2017 (Springer).
22. Dokmai, N., Kockan, C., Zhu, K., Wang, X., Sahinalp, S.C., and Cho, H. (2021). Privacy-preserving genotype imputation in a trusted execution environment. Cell Syst. *12*, 983–993.e7.
23. Gürsoy, G., Chielle, E., Brannon, C.M., Maniatakos, M., and Gerstein, M. (2022). Privacy-preserving genotype imputation with fully homomorphic encryption. Cell Syst. *13*, 173–182.e173.
24. Kim, M., Harmanci, A.O., Bossuat, J.-P., Carpov, S., Cheon, J.H., Chillotti, I., Cho, W., Froelicher, D., Gama, N., Georgieva, M., et al. (2021). Ultrafast homomorphic encryption models enable secure outsourcing of genotype imputation. Cell Syst. *12*, 1108–1120.e4.
25. Grishin, D., Obbad, K., and Church, G.M. (2019). Data privacy in the age of personal genomics. Nat. Biotechnol. *37*, 1115–1117.
26. Malakar, Y., Lacey, J., Twine, N.A., McCrea, R., and Bauer, D.C. (2024). Balancing the safeguarding of privacy and data sharing: perceptions of genomic professionals on patient genomic data ownership in Australia. Eur. J. Hum. Genet. *32*, 506–512.
27. Kuo, T.-T., Jiang, X., Tang, H., Wang, X., Bath, T., Bu, D., Wang, L., Harmanci, A., Zhang, S., Zhi, D., et al. (2020). iDASH secure genome analysis competition 2018: blockchain genomic data access logging, homomorphic encryption on GWAS, and DNA segment searching. BMC Med. Genomics *13*, 98.
28. Gürsoy, G., Brannon, C.M., Ni, E., Wagner, S., Khanna, A., and Gerstein, M. (2022). Storing and analyzing a genome on a blockchain. Genome Biol. *23*, 134.
29. Grishin, D., Raisaro, J.L., Troncoso-Pastoriza, J.R., Obbad, K., Quinn, K., Misbach, M., Gollhardt, J., Sa, J., Fellay, J., Church, G.M., and Hubaux, J. P. (2021). Citizen-centered, auditable and privacy-preserving population genomics. Nat. Comput. Sci. *1*, 192–198.
30. Siva, N. (2008). 1000 Genomes project. Nat. Biotechnol. *26*, 256–257.
31. Yoshida, A., Huang, I.-Y., and Ikawa, M. (1984). Molecular abnormality of an inactive aldehyde dehydrogenase variant commonly found in Orientals. Proc. Natl. Acad. Sci. USA *81*, 258–261.
32. Ellis, J.A., Stebbing, M., and Harrap, S.B. (2001). Polymorphism of the androgen receptor gene is associated with male pattern baldness. J. Invest. Dermatol. *116*, 452–455.
33. Sherry, S.T., Ward, M.-H., Kholodov, M., Baker, J., Phan, L., Smigielski, E. M., and Sirotkin, K. (2001). dbSNP: the NCBI database of genetic variation. Nucleic Acids Res. *29*, 308–311.
34. De Canniere, C., and Preneel, B. (2008). Trivium. In New Stream Cipher Designs: The eSTREAM Finalists, M. Robshaw and O. Billet, eds. (Springer), pp. 244–266.
35. Ben-Sasson, E., Chiesa, A., Genkin, D., Tromer, E., and Virza, M. (2013). SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge (Springer), pp. 90–108.
36. Gentry, C. (2009). Fully Homomorphic Encryption Using Ideal Lattices. STOC '09: Proceedings of the forty-first annual ACM symposium on Theory of computing. https://doi.org/10.1145/1536414.1536440.
37. Hail | GWAS Tutorial. https://hail.is/docs/0.2/tutorials/01-genome-wide-association-study.html.
38. Butler, J.M. (2006). Genetics and genomics of core short tandem repeat loci used in human identity testing. J. Forensic Sci. *51*, 253–265.
39. Du, Z. (2000). Polymorphism of 13 STR loci for establishment of Chinese criminal DNA database. Fa yi xue za zhi *16*, 63–65.
40. Check Hayden, E. (2017). The rise and fall and rise again of 23andMe. Nature *550*, 174–177.
41. Chillotti, I., Gama, N., Georgieva, M., and Izabachène, M. (2020). TFHE: fast fully homomorphic encryption over the torus. J. Cryptol. *33*, 34–91.
42. Smart, N.P., and Vercauteren, F. (2014). Fully homomorphic SIMD operations. Des. Codes Cryptogr. *71*, 57–81.
43. Keller, M. (2020). MP-SPDZ: A Versatile Framework for Multi-Party Computation. CCS '20: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. https://doi.org/10.1145/3372297.3417872.
44. Canteaut, A., Carpov, S., Fontaine, C., Lepoint, T., Naya-Plasencia, M., Paillier, P., and Sirdey, R. (2018). Stream ciphers: A practical solution for efficient homomorphic-ciphertext compression. J. Cryptol. *31*, 885–916.
45. Asharov, G., Jain, A., López-Alt, A., Tromer, E., Vaikuntanathan, V., and Wichs, D. (2012). Multiparty Computation with Low Communication, Computation and Interaction via Threshold FHE (Springer).
46. Cramer, R., Damgård, I.B., and Nielsen, J.B. (2015). Secure Multiparty Computation and Secret Sharing (Cambridge University Press).
47. Goyal, V., Li, H., Ostrovsky, R., Polychroniadou, A., and Song, Y. (2021). ATLAS: Efficient and Scalable MPC in the Honest Majority Setting (Springer).
48. Keller, M., Orsini, E., and Scholl, P. (2016). MASCOT: Faster Malicious Arithmetic Secure Computation with Oblivious Transfer. CCS '16: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. https://doi.org/10.1145/2976749.2978357.
49. Albrecht, M., Grassi, L., Rechberger, C., Roy, A., and Tiessen, T. (2016). MiMC: Efficient Encryption and Cryptographic Hashing with Minimal Multiplicative Complexity (Springer).
50. Chillotti, I., Joye, M., and Paillier, P. (2021). Programmable Bootstrapping Enables Efficient Homomorphic Inference of Deep Neural Networks (Springer).
51. Regev, O. (2010). The Learning with Errors Problem (Invited Survey). IEEE 25th Annual Conference on Computational Complexity. https://doi.org/10.1109/CCC.2010.26.
52. Botrel, G., Piellard, T., Housni, Y., Kubjas, I., and Tabaie, A. (2023). ConsenSys/gnark: v0.8.0. https://github.com/consensys/gnark/tree/v0.8.0.
53. Sikorska, K., Lesaffre, E., Groenen, P.F.J., and Eilers, P.H.C. (2013). GWAS on your notebook: Fast semi-parallel linear and logistic regression for genome-wide association studies. BMC Bioinf. *14*, 1–11.
54. Goldschmidt, R.E. (1964). Applications of Division by Convergence (Massachusetts Institute of Technology).

CellPress
OPEN ACCESS

STAR★METHODS

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Deposited data		
Source data of Governome	This paper	http://www.bio8.cs.hku.hk/governome/
1000 Genomes	1000 Genomes Project Consortium[30]	PRJNA262923
Software and algorithms		
Governome	This paper	https://github.com/HKU-BAL/Governome; https://zenodo.org/records/16780081
Gnark	Gnark team	https://github.com/Consensys/gnark
TFHE-go	Intak Hwang	https://github.com/sp301415/tfhe-go
MP-SPDZ	Marcel Keller[43]	https://github.com/data61/MP-SPDZ

METHOD DETAILS

Feasible approaches to fulfill the three properties of owner-governance

In this section, we discuss the techniques used in Governome and how they serve to fulfill the three properties of owner-governance. As shown in Figure 1, Governome comprises three layers: a consensus layer for authority management, a computing layer for secure computation, and an application layer to make use of the other two layers for genomic applications. At the consensus layer, Blockchain is used to enable dynamic permission control, and Zero-knowledge Proof is used to enforce data integrity. At the computing layer, cryptographic techniques, including Homomorphic Encryption, stream cipher, and Secure Multiparty Computation, are used to fulfill secure computation. At the application layer, several design focuses are introduced to establish the fundamental guidelines for building an efficient owner-governed genomic data management system.

Techniques used at the consensus layer

Why use Blockchain?. Owner Authority strictly requires decentralization, as centralization would technically inevitably jeopardize owners' control of their data despite how many non-technical promises have been made. Blockchain, as a proven decentralized solution that can achieve consensus, is considered a natural choice for an owner-governed genomic data management system. Blockchain can be configured to ensure that no party can exercise power on others' data except for their own. Due to the transparent and traceable nature of blockchain, data owners can access their consent and data usage logs at any time, thus ensuring auditability. Blockchain can also be used to enforce consensus on computation results so fraud by minorities can be avoided. Both public blockchain and private blockchain are applicable to Governome. While renowned public blockchains are trusted for their decentralization and diversification of users, hence more suitable for publicly or internationally initiated genome hosting, a private blockchain is more flexible and cost-effective for locally initiated genome hosting, in which decentralization is less of a concern.

Why use Zero-knowledge Proof?. Life cycle Data Encryption requires genomic data to remain encrypted throughout its life cycle in the system. Ciphertext at both storage and computation makes it hard if not impossible to avoid tampering or fraud through traditional means, such as revealing the data or computation results for public scrutiny. In order to ensure the genomic data and computation results are not tampered with or frauded, we use zero-knowledge proof. Zero-knowledge proof[35] allows a prover to generate a proof for a proposition without revealing any of its input. In Governome, any data loaded and stored is encrypted with a stream cipher, meanwhile a stream cipher key (SCK) is generated and held by the data owner. When a data owner needs to prove she is providing untampered data, Governome uses Zero-knowledge proof to prove that she is providing an encryption of the right SCK to make genomic data accessible without revealing any part of the SCK. Tempered SCK will lead to a different hash that mismatches what has been saved on-chain, thus failing the proof.

Techniques used at the computing layer

Why use Homomorphic Encryption?. The only way to ensure no data leakage is either no data to leak or leakage doesn't matter. In Governome, LDE mandates that no plaintext exists in the system. Thus, a solution that supports verifiable computation with ciphertext only is needed. Homomorphic Encryption (HE)[36] is such a solution that produces deterministic computation results verifiable by all users in the system, using only encrypted data and requiring zero decryption. In contrast, hardware-based solutions, like Intel SGX (Software Guard Extensions) and AMD Memory Encryption Technology, cannot fulfill LDE because they require data to be decrypted when exiting the hardware that supports the same solution. They also cannot fulfill VER because their computational results cannot be easily verified by users who lack the same hardware solution. The details about the HE schemes used in Governome can be found in the 'homomorphic encryption scheme' subsection in STAR Methods.

Why use a stream cipher?. We use HE to fulfill LDE in Governome. However, conversion from plaintext to HE ciphertext (ciphertext capable of HE-based computations) expands the data size by over three orders of magnitude,[41] making it inefficient if not impossible to store HE ciphertext. In Governome, we solve the problem by encrypting plaintext with a cipher that 1) does not significantly increase the size of ciphertext so the ciphertext can be stored efficiently, and 2) can convert from ciphertext to HE ciphertext on-the-fly and without decryption for analysis. Stream cipher[44] fulfills these requirements. In Governome, with the use of stream cipher, stream cipher ciphertexts are stored, and will be converted to temporary HE ciphertexts in secure domain when analysis needs them. More details about how genomic data is encrypted and saved, along with how the stream ciphertext is transferred into HE ciphertext are available in the 'storage and computation setup in governome' subsection in STAR Methods.

Why require multiple parties for the generation of HE key?. Genomic data that are converted into HE ciphertext can be used for analysis, and as the nature of HE, the computation results are also in HE ciphertext. Inevitably, the results are required to be decrypted to become readable before leaving the system. The decryption requires the key that was used to encrypt HE ciphertext. However, that implies that anyone who holds the complete key can decrypt the HE ciphertext to obtain the original genomic data or computation results. In Governome, we 1) required a collaborative generation of a complete key, and 2) avoided any single parties having a copy of the complete key. Our solution uses Threshold Fully Homomorphic Encryption[45] (ThFHE), which is a type of Secure Multiparty Computation[46] (SMPC). SMPC enables multiple parties to collaborate to generate a key without disclosing the input of any parties, and the protocol can run smoothly to a consensus even if one-third of the participants are dishonest. Furthermore, SMPC can be used not only for key generation but also for HE ciphertext decryption. So, without revealing the complete key to any party, SMPC then uses the key and the multiple parties who generated the key to decrypt the computation results. The correctness of computation results is ensured by the SMPC protocol.[43,47,48] If not using SMPC, one might think of isolation measurements such as limiting the interactions between computing parties (that do compute but are not eligible to see the results) and an HE key holder (that are eligible to see the results), so the computing parties cannot see any intermediate results without the key. However, compliance with such measurements is not algorithmically guaranteed, which is against VER in Governome.

Design focuses at the application layer

The consensus layer and computing layer together enable owners to have around-the-clock full governance and security of their data. The application layer, on the other hand, is about how to make use of genomic data. An application layer should 1) provide necessary but minimal functions to accomplish different genomic analysis tasks; 2) be straightforward for query entities to use; and 3) operate efficiently with a clear timetable. The design of the application layer of Governome has the following major focuses.

Functionality Minimization. The application layer specifies permissible actions on existing genomic data through a set of functions. These functions must be carefully reviewed by the research ethics community to remain essential yet minimal. Once a function is recorded on the public blockchain, it becomes immutable, accessible, and verifiable by everyone. Given that there is little consensus on whether some genomic applications are inherently safe, emphasizing interpretability and minimalism is advisable.

Interaction Simplification. Interactions between query entities and the application layer should be straightforward. Typically, query entities are doctors and researchers, who may have clinical and research expertise but a lack deep understanding of privacy-enhancing technologies. Therefore, a clear and concise interface is essential to help them engage with genomic data in a familiar manner.

Clear Timetable. Applicants often encounter lengthy procedures that are even more time-consuming than data mining when applying for data from government-funded databases.[6–8] To ensure transparency and reasonableness, it is therefore necessary to maintain a clear timetable for preset applications. Generally, query entities should be able to receive timely responses, including logs and result output.

Storage and computation setup in Governome

How to encrypt genomic data

In Governome, raw genomic data is encrypted with stream ciphers and stored in distributed storage nodes that are organized by a blockchain. As data owners have around-the-clock full governance of their data, they are supposed to hold the SCK and are being asked for it every time their data is being used for computing. However, a practical concern is that data owners might leak their SCK due to incidents such as device loss or data theft. The risk is accumulative and gets more significant when the sample size increases. To address the issue, Governome uses hospitals as an additional SCK holder. Instead of holding a complete SCK, data owners and hospitals each hold only a part of the key. Governome collects the two partial access tokens (HE ciphertext form of the two partial keys) from the data owner and hospital, and recovers the full access token (HE ciphertext form of the complete SCK) with secure computation supported by HE (see Figure S1). Details about how genomic data is segmented and stored are described in the 'data segmentation' subsection in STAR Methods, and the discussion of security concerns can be found in the 'security of the data blocks' subsection in STAR Methods. This design reduces the risk of leaking the complete SCK. Noteworthy, although hospitals also hold part of the data owner's SCK, they have no right over the data owner's genomic data because in Governome, data ownership is ascertained through consensus on the blockchain rather than through the possession of SCK.

How computing layer works

The computing layer in Governome involves multiple parties for managing and using stream ciphertext and HE ciphertext, as shown in Figure S2. The stream cipher we have chosen is Trivium,[34] a symmetric encryption scheme in which the decoding algorithm consists only of Boolean operations. Through an XOR operation in the secure domain, the full token (HE ciphertext of the stream cipher

key) is recovered from the two partial tokens collected from the data owner and hospital. Next, the stream ciphertext is encrypted with HE. At this point, the data is double encrypted using both stream and HE ciphers. The reason why we cannot decrypt the stream ciphertext before encrypting it into HE ciphertext is because Governome does not allow fully decrypted data at any point. Then, the double encrypted data is decrypted through HE operations to remove the stream cipher layer. The result of the decryption is HE ciphertext and can be used for subsequent HE-based computations. The computing layer can carry out different genomic analysis tasks according to the query. The generation of HE keys uses SMPC, and therefore no single computing node has the complete copy of it, eliminating the chance that a single computing party could peek or tamper with the HE key. The analysis results are in encrypted form and will be decrypted using SMPC before returning to a query entity.

Precomputed access token

When a data owner's data is asked to be included for analysis, she is requested to submit an access token generated from her SCK and a Governome-given HE key not only for her data to be used for HE-based computing, but also as a gesture of granting access. However, this behavior requires data owners to respond actively; otherwise, their data would not be included for analysis. This requirement might be too demanding for some data owners who are always willing to be involved in analyses as long as their privacy is protected. Precomputed access tokens are such a mechanism in Governome to allow a sharing data owner to register a precomputed access token for accessing all her data blocks (see the 'data segmentation' subsection in STAR Methods) so that she does not need to respond to Governome for their data to be used.

Necessary supporting parties in Governome

In Governome, besides data owners, multiple parties with different roles are involved to form a robust genomic data analysis system. Their duties and importance are explained as follows.

(1) Hospitals or institutions that are eligible to host demographic and phenotypic data: In Governome's design, it hosts only genotypic data and does not host phenotypic data. This is an effective measurement to 1) isolate different types of critical data, and 2) avoid a single party getting over-powerful. The participating hospitals and institutions connect to the blockchain. Their communications are encrypted and algorithmically verifiable. If a query is asking for a specific cohort with demographic or phenotypic constraints, Governome will ask each participating hospital or institution to provide a list of qualified and anonymous sample IDs. Hospitals and institutions are allowed to return an incomplete (or even empty) list of qualifying samples because they also have the power that equals the Governome to refuse individual data usage. The sample IDs are anonymous by using data owners' blockchain address.

(2) Super users that will never withdraw from the system: Both the consensus layer and computing layer of Governome require multiple active users to maintain functioning. While data owners are granted full governance of their data in Governome, few of them might be active users that can host the blockchain and support the computing in Governome. Super users are a group of users that run servers and can provide storage (i.e., storage nodes) and computing resources (i.e., computing nodes) to keep a Governome system running. Within the data usage life cycle of genomic data in Governome, super users are responsible for checking hashes and proofs to ensure correctness, pulling data from the storage nodes, converting data into HE ciphertext, and performing the actual computations. The computation results are ultimately decrypted collaboratively by the super users and returned to the query entities through the interface of the application layer. Super users are usually academic institutions, governmental authorities, hospitals, and pharmaceutical companies - the major stakeholders in the system who will mostly benefit from a stable and growing Governome system. In Governome, we require two or more super users to be involved in security-critical procedures, including SMPC, HE key generation, and computation results decryption. While super users have a higher responsibility to keep a Governome system operational, they have data usage privileges identical to all data owners.

(3) Temporary computing nodes that temporarily provide additional computing power. Large-scale cohort studies and GWAS analyses are usually conducted by institutional users who are willing to contribute temporary computing nodes in return for some speed up in obtaining a result.

Data structure and cryptographic algorithms in Governome

Data segmentation

Genomic analysis tasks that query variants usually request one or a few variants. Therefore, a data segmentation design that optimizes the query of a small number of variants, while having minimal overhead when retrieving many or all variants is required.

In Governome, we use hash indexing to partition variants into data blocks. The number of data blocks per individual, denoted as b, is configurable but unchangeable after system initiation. We suggest storing an average of 40–50 variants per data block. For example, an individual with whole-genome sequenced usually has four to five million variants, thus b = 96k is an optimal setting. When using Governome to store only exome sequencing variants, b should be reduced. A principle is that the average number of variants per data block should not go below 20 to avoid data leakage through probing (More details are shown in the subsection 'Security of data blocks'). The storage size of each variant is 36 bits, with 32 bits for rsID and four bits for genotype (16 possibilities including 0|0, 0|1, 0|2, 0|3, 1|0, 1|1, 1|2, 1|3, 2|0, 2|1, 2|2, 2|3, 3|0, 3|1, 3|2, and 3|3). For each variant, the corresponding data block ID is calculated as SHA3($sampleID$,$rsID$) mod b. $sampleID$ is a string no longer than 15 characters (e.g., "NA12878" in 1kGP). In practice,

the *sampleID* can be the user ID of the data owner or her corresponding blockchain address with a random suffix (salt). We used SHA3 as the hash function, which is provided by the Ethereum blockchain as a built-in function that runs efficiently on-chain.

A data block is the smallest data unit to be retrieved for a query in Governome. Each data block is encrypted with a different Stream Cipher Key (SCK). Each SCK only gives access to a data block. So, multiple SCKs are needed to access multiple or all variants of an individual. This design enables Governome to support fine-grained access control, giving data owners precise control over who can access which data blocks. However, it is not necessary for a data owner to really hold multiple SCKs. In fact, the storage and access of tens of thousands of SCKs is computationally inefficient. In Governome, each data owner holds only a primary SCK. The SCK for each data block is derived from the primary SCK using $MiMC(\text{primary SCK}, segmentID)$. We used MiMC[49] as the hash function here, which is efficient in the ZK-SNARKs circuit.

As elaborated in the 'Storage setup in Governome' subsection of Methods, the SCKs held by data owners and hospitals are partial SCKs. To recover a complete SCK, a data owner and her responsible hospital both need to provide their corresponding partial SCK. A complete SCK is calculated as $SCK_{dataOwner} \oplus SCK_{hospital}$, where $SCK_{dataOwner} = MiMC(\text{primary } SCK_{dataOwner}, segmentID)$, and $SCK_{hospital} = MiMC(\text{primary } SCK_{hospital})$. Notice that hospital uses primary SCK for all data blocks with the rationale that hospital will reject data access to an individual as a whole, instead of rejecting access to particular data blocks.

Security of the data blocks

An observation in human genomics is that the number of variants of a normal human is only 0.1–0.2% of the genome size. Thus, as a storage-efficient design, in each data block, only variants with alternative alleles are stored (i.e., variants with the reference allele are not stored). However, that enables an attacker to do probing, that is, to inspect the size of a data block to determine the number of variants in the data block. Although the attacker wouldn't be able to decrypt the data block without the correct SCK, it is possible that one can infer whether a variant exists or not in a person if the data block has only one or few variants and if the assignment of a rsID is the same for each individual.

To address the concerns, firstly, the assignment of a rsID (to which data block) is unique for each individual in Governome by adding salt. Salt is an individual unique value that is added to the input of a hash function to create unique hashes for every input. Governome uses the random suffix in *sampleID* as the individual-unique value.

Secondly, even with a sensible setting of b, a small number of data blocks will still contain fewer than 20 variants according to the normal distribution. We solved the problem by adding space fillers to a data block to ensure each data block is 20 or more variants long in size. As data blocks are stored in stream ciphertext, attackers wouldn't know how many space-fillers are in a data block. The minimum number of variants per data block, 20, can be raised for better security, but at the price of more wasted storage.

Auxiliary data block

Governome allows auxiliary data blocks besides general data blocks. Auxiliary data blocks are task-specific and aim to provide better storage and computation efficiency for specific tasks.

Using forensics analysis as an example. The design of general data blocks that randomly disperses variants into multiple data blocks with individual-specific hashing implies that retrieving n random variants will most likely need to retrieve n data blocks. This design was optimized for the best security when the variants of each individual can vary a lot and use only a small subset of all available rsID (otherwise, storing all rsID would be a better solution). However, for forensics analysis that 1) inspects only a fixed set of about tens of variants per individual, and 2) needs to scan over a large number of individuals, the overhead of using general data blocks becomes significant and caps the scale of Governome. Auxiliary data blocks address the issue by storing the variants required by a task in a single fixed-size data block. Thus, for forensics analysis, only one data block is needed to be retrieved per individual.

In Governome, as a proof of concept, we used an auxiliary data block design that stores 13 commonly used Short Tandem Repeats (STRs),[39] including D3S1358, vWA, FGA, D8S1179, D21S11, D18S51, D5S818, D13S317, D16S539, THO1, TPOX, CSF1PO, and D7S820. The list can be of any size to accommodate different types of tasks. Multiple auxiliary data block designs are allowed in Governome, but the designs are immutable after system initiation.

Homomorphic encryption scheme

So far, the majority of homomorphic encryption used in bioinformatics practices refers to second-generation homomorphic encryption. As computational complexity increases, the efficiency of second-generation homomorphic encryption technology drastically declines, rendering it impractical in some cases. For instance, while second-generation homomorphic encryption supports GWAS, it is limited to pre-processed data and does not support preprocessing tasks such as data cleaning and quality control. In contrast, third-generation homomorphic encryption does not have these limitations and is considered the best practice for deep neural networks in the era of machine learning. Therefore, we have chosen third-generation homomorphic encryption as the primary technology for the computing layer in Governome. In this section, we review the technical details of the third-generation homomorphic encryption scheme used in Governome.

For better flexibility, in Governome, we chose the third-generation homomorphic encryption scheme TFHE,[41] which is based on torus. TFHE computations occur within a small integer domain T and support two types of operations: addition operations and programmable bootstrapping.[50] Addition operations include negation and simple addition, through which we can obtain linear combinations of known ciphertexts. Programmable bootstrapping is a single-variable operation that allows us to construct and implement a single-variable mapping $f : T \rightarrow T$, where f can be arbitrarily chosen. In Governome, we construct equivalent Boolean circuits based on these two operations to perform computations.

The hardness of TFHE is based on LWE (Learning With Errors[51]). The encryption process is as follows.

(1) $KeyGen : \boldsymbol{s} \leftarrow Z_q^n, pk \leftarrow (\boldsymbol{A}, \boldsymbol{b}), where\ \boldsymbol{A} \leftarrow Z_q^{n \times n}, \boldsymbol{b} = \boldsymbol{As} + e$
(2) $Encrypt_{pk}(m) : ct = (a^*, b^*), where\ r \leftarrow \{0,1\}^n, \boldsymbol{a}^* = \boldsymbol{Ar} + \boldsymbol{e_1}, b^* = \langle \boldsymbol{b}, \boldsymbol{r} \rangle + \Delta m + e_2$
(3) $Decrypt_s(ct) : return\ \boldsymbol{round}(b^* - \langle \boldsymbol{a}^*, \boldsymbol{s} \rangle)$

Additionally, to achieve ThFHE and conceal the HESK (Homomorphic Encryption Secret Key), Secure Multi-parties need to collaborate to generate some auxiliary information based on the unseen HESK. This includes: 1) HEPK (Homomorphic Encryption Public Key) for encryption. 2) KSK (Keyswitch Key) and BSK (Bootstrap Key) for computation. 1 will be made public, allowing anyone to encrypt their information using it. 2 will be submitted to the computing nodes; otherwise, the computation cannot be performed.

Noteworthy, the unseen HESK and the aforementioned 1 and 2 need to be updated periodically. Otherwise, even if the data owner revokes sharing, the access tokens they have submitted can always be accessed by attackers.

Zero-knowledge proof for access token generation

In order to ensure data integrity, data owners and hospitals need to prove that they have submitted the correct access token without revealing any part of it. To achieve this, we use ZK-SNARK, which is a framework of zero-knowledge proof that allows a prover to generate a proof for a proposition without revealing any of its input. In this section, we will illustrate how an access token and its corresponding zero-knowledge proof is generated.

When a genomic analysis initiates in Governome and the public blockchain asks for access tokens from the corresponding data owners and hospitals, what is collected from the data owners and hospitals are partial access tokens. A partial access token is an encrypted partial SCK encrypted with HEPK, plus an honesty proof to enforce data integrity. The partial SCK is generated from the primary SCK, whose hash is stored on-chain as a credential. A usual data owner usually uses a smartphone or laptop to interact with Governome. The 'generating proof for access token' subsection in Results showed that generating a ZK-SNARK proof is a memory demanding procedure. Therefore, we chose to split the partial SCK into smaller blocks to use and reuse smaller zero-knowledge proof circuits that fit smaller blocks to reduce memory consumption. The pseudocode of ZK-SNARKs is shown below, with the use of library gnark.[52]

Pseudocode for ZK-SNARKs.

```
Input:
    Public variables pk,blocksize
    Public witness blockID,segID,credential,ciphertext
    Private witness primarySCK,error
Procedure:
    mask←(1<<blocksize)-1
    block←hash(primarySCK,segID,blockID) & mask
    assert hash(primarySCK)=credential
    for i from 0 to blocksize-1 do
      bit←block & 1
      ct←LWE(bit,error[i],pk)
      assert error[i] is valid
      assert ct=ciphertext[i]
      block←block>>1
    generating a proof π for the assertions
```

As shown in the pseudocode above, we split the partial SCK into length/blocksize blocks and generate proofs for each block sequentially. The original single proof is divided into length/blocksize proofs, thus the scale of the proof circuit can be significantly reduced. The "Generating proofs for an access token" subsection in Results shows that when blocksize is set to 1, the memory consumption is 1.1GB, which can be accommodated by most of the modern laptops.

Moreover, zero-knowledge proof requires even better security regarding the hash function, so instead of using SHA3 that was used in data segmentation and indexing, we used MiMC,[49] which is based on large prime fields as the hash function in zero-knowledge proof.

HE-based GWAS analysis

In this section, we describe the details about HE-based GWAS analysis that calculates the *p*-value of individual variants against a phenotype using linear regression.[53] Since the algorithm is implemented in a Boolean circuit, division and floating-point operations

are very expensive. Therefore, it is necessary to perform transformations on the GWAS formula to convert most floating-point operations in linear regression to integer operations, and to minimize the number of divisions.

To be specific, the linear regression model is defined as follows:

$$y = \beta_0 + \beta_1 x + \varepsilon$$

Here, y is the phenotype, while x is the genotype. In our benchmark, we used CaffeineConsumption (1 if CaffeineConsumption>4, 0 otherwise) as y, and x as either 0, 1, or 2 (reference, heterozygous variant, homozygous variant).

According to the definition and formula of linear regression, we have:

$$\widehat{\beta_0} = \frac{\Sigma y_i \Sigma x_i^2 - \Sigma x_i \Sigma x_i y_i}{n\Sigma x_i^2 - (\Sigma x_i)^2}$$

$$\widehat{\beta_1} = \frac{n\Sigma x_i y_i - \Sigma x_i \Sigma y_i}{n\Sigma x_i^2 - (\Sigma x_i)^2}$$

To perform a t-test, we need to calculate the t-value as follows:

$$t = \frac{\widehat{\beta_1}}{s_{\hat{\beta}_1}} \sim t_{n-2}$$

where $s_{\hat{\beta}_1} = \sqrt{\left(\frac{\Sigma(\widehat{\beta_0}+\widehat{\beta_1}x_i - y_i)^2}{(n-2)\Sigma(x_i-\bar{x})^2}\right)}$

In the above formula, the sample size n is known, and covariance calculation is not needed. Once we calculated the t-value, the *p*-value can be calculated using the cumulative distribution function.

Furthermore, we can perform the following transformation on the above formula:

$$\frac{n}{n-2} \cdot t^2 = \frac{Numerator}{Denominator},$$

where *Numerator* and *Denominator* can be written as:

$$Numerator = (n\Sigma x_i y_i - \Sigma x_i \Sigma y_i)^2 \left(n\Sigma x_i^2 - (\Sigma x_i)^2\right),$$

$$\begin{aligned} Denominator = {} & n(\Sigma x_i y_i)^2 (\Sigma x_i)^2 - n\left(\Sigma x_i^2\right)^2 (\Sigma y_i)^2 - n^2 \Sigma x_i^2 (\Sigma x_i y_i)^2 + \Sigma x_i^2 (\Sigma x_i)^2 (\Sigma y_i)^2 + n^2 \left(\Sigma x_i^2\right)^2 \Sigma y_i^2 - 2n\Sigma x_i^2 \Sigma y_i^2 (\Sigma x_i)^2 \\ & + \Sigma y_i^2 (\Sigma x_i)^4 + 2n\Sigma x_i^2 \Sigma x_i y_i \Sigma x_i \Sigma y_i - 2\Sigma x_i y_i (\Sigma x_i)^3 \Sigma y_i. \end{aligned}$$

With the transformation, we can calculate the *Numerator* and *Denominator* through HE-based computation. Subsequently, by truncating the *Numerator* and *Denominator* to 16-bit precision and performing Goldschmidt division,[54] we can obtain an accurate t-value, which is then converted to a *p*-value in plaintext operations. Notably, while covariance calculation is not needed, the calculation of *p*-value of each variant is independent, thus can be processed in parallel.

QUANTIFICATION AND STATISTICAL ANALYSIS

Zero-knowledge proof generation with gnark

To evaluate the performance of zero-knowledge proofs, we divided the 80-bit stream cipher key into blocks and sequentially generated proofs for each block. The memory consumption was measured by recording the peak memory usage during the generation of a single proof, while the time consumption was recorded by measuring the total time taken to generate all proofs.

HE-based computation with TFHE-go

To assess the performance of HE-based computation, we independently measured the time taken for data conversion and data analysis across different cohorts. In data conversion, the process begins with the verification of a stream cipher key attached by a zero-knowledge proof and ends when all the necessary data for downstream analysis has been converted into HE ciphertext. In data analysis, the time is recorded until a valid output is obtained, which includes a genotype in an individual variant query, the count of valid individuals in querying a cohort, a t-value derived from a Student's t test in GWAS, or a list indicating whether an individual matches the target STR in forensics.