When Gene Meets Cloud: Enabling Scalable and Efficient Range Query on Encrypted Genomic Data

Wenhai Sun, Ning Zhang, Wenjing Lou, and Y. Thomas Hou
Virginia Polytechnic Institute and State University, Blacksburg, VA, USA

***Abstract*—As the cost of human full genome sequencing continues to fall, we will soon witness a prodigious amount of human genomic data in the public cloud. To protect the confidentiality of the genetic information of individuals, the data has to be encrypted at rest. On the other hand, encryption severely hinders the use of this valuable information, such as Genome-wide Range Query (GRQ), in medical/genomic research. While the problem of secure range query on outsourced encrypted data has been extensively studied, the current schemes are far from practical deployment in terms of efficiency and scalability due to the data volume in human genome sequencing.**

In this paper, we investigate the problem of secure GRQ over human raw aligned genomic data in a third-party outsourcing model. Our solution contains a novel secure range query scheme based on multi-keyword symmetric searchable encryption (MSSE). The proposed scheme incurs minimal ciphertext expansion and computation overhead. We also present a hierarchical GRQ-oriented secure index structure tailored for efficient and large-scale genomic data lookup in the cloud while preserving the query privacy. Our experiment on real human genomic data shows that a secure GRQ request with range size 100,000 over more than 300 million encrypted short reads takes less than 3 minutes, which is orders of magnitude faster than existing solutions.

I. Introduction

We are on the cusp of a new era in genomic research. Full genome sequencing (FGS) is conducted to comprehensively decode an organism's genetic make-up. It allows us to gain an unprecedented level of understanding on the biological inner workings. Medical/genomic researchers can now predict disease susceptibilities and drug responses base on a person's genome. Over the past decade, the cost for sequencing the genome of a person has been substantially reduced from $\$100$ million or so in 2001 to roughly $\$1,000$ in 2015 [1]. The promising future of personalized medicine is now within reach due to the sequencing cost reduction. On the other hand, one of the questions we need to answer is how one can effectively and efficiently handle the data proliferation as a result of FGS [2]. Large volume of genomic data from patients becomes a challenge for medical facilities. Public cloud, such as Google Genomics, DNAnexus, etc., may provide us with an economical solution for storage, but it also exposes users to the pitfall of privacy breach [3]. Due to the "highly-sensitive" nature of genomic data, such data loss can lead to severe consequences.

Secure Genome-wide Range Query. The human genome is composed of sequences of nucleotides. The process of human genome sequencing records segments of these sequences as short reads. Each short read contains genetic information of a biological property of an individual. Short reads are aligned to the human reference genome to determine its relative position in the chromosome. In medical research, scientists and medical professionals obtain short reads within a specific range in the chromosome to study the genetic attributes of an individual. This search is often referred to as *Genome-wide Range Query*. It retrieves all the short reads of interest *within the queried range* that is defined by a lower and upper position in *the entire genome*. Usually, such study is supervised or regulated by a Trusted Authority, such as National Institutes of Health (NIH), Food and Drug Administration (FDA). GRQ has been widely adopted in many applications in real world. For instance, a pharmaceutical company hopes to issue a query on a particular range of genomic data of a patient in order to find some DNA fingerprints/biomarkers for personalized medicine. In this case, the company needs to obtain approval on such query from the FDA first [4], [5]. For another example, an authorized physician can request a range of your genome for certain genetic tests that may indicate your potential response to a drug before he makes a clinical decision. With the low-cost DNA sequencing for the masses, Genome-wide Range Query given in the above examples is expected to be extensively used on top of the genomic data storage in many healthcare-related services and genomic research [6], [7].

Intuitively, existing secure range query techniques, e.g., order-preserving symmetric encryption (OPSE) [8], predicate encryption (PE) [9], [10], etc., may be the promising building blocks for a secure GRQ realization. However, directly applying them to an enormous amount of genomic data will cause privacy and efficiency concerns. In general, genomic data is range-sensitive because given specific query range information, the adversary can easily figure out the underlying genetic test. In order to facilitate efficient query, possible solutions [8], [10] in the literature may outright leak data ordering information which can be employed to compromise the range privacy. On the other hand, naïve use of current raw genomic data structure for multiprocessing will result in a significant scalability issue in the cloud [11], [12]. Direct adoption of the "heavy" cryptographic tools or an ill-designed index structure for the cloud deployment will even worsen the situation by introducing prohibitive performance penalty.

Our Contributions. In this paper, our several key observations enable a novel solution to the efficient and scalable secure GRQ design. We first observe that the current secure range query solutions, when used in the context of GRQ, suffer from either slow search process [9], or compromised

security guarantees [8], [10], [13], i.e. fully revealing ordering information, so as to achieve efficient query, let alone their scalability constraint. Straightforward as this order-comparison method is, it also gives the adversary a distinct advantage in the range identification with a reference genome.

We propose a novel secure range query scheme by designing a multi-keyword symmetric searchable encryption, which may be of independent interest. It is suitable for search over space-consuming raw genomic data storage. The security of the proposed scheme is derived from the MSSE security against adaptive chosen-keyword attacks (CKA2) [14]. Our scheme shows the same privacy guarantees as the current practical solution but with a much better (logarithmic-time) query efficiency.

By plaintext ordering obfuscation and a privacy-preserving re-sorting process, we present a hierarchical secure index structure, which captures the real-world situation of genomic data processing in the cloud, featuring a scalable and efficient GRQ search over human raw aligned genomic data.

We implement our proposed scheme on real human sequencing data. The experiment demonstrates the efficiency, and its promising future deployment in a large-scale GRQ scenario.

II. Related Works

Cryptographic Range Query. Boldyreva *et al.* proposed an order-preserving symmetric encryption [8] that allows the ordering of the numerical plaintexts to be preserved in their encrypted forms. As a result, range query can be realized by ciphertext comparison. Apart from the practical security concerns [15], due to its order-revealing property, OPSE is not an appropriate building block for GRQ, which enables the adversary to determine the query range information trivially. Boneh *et al.* proposed an order-revealing symmetric functional encryption scheme [13], which is still inefficient for practical use due to the computationally expensive multilinear map operations. It also suffers from the order privacy breach as OPSE in the GRQ scenario.

Predicate encryption is another promising cryptographic primitive for GRQ. The decryption will succeed should the ciphertext fall into the queried range. In the public key setting, Shi *et al.* [16] proposed a PE scheme for multidimensional range query with linear search complexity. Note that the major inhibitor for the adoption of asymmetric PE in practice is the *predicate privacy* breach [9], i.e., an adversary can generate a ciphertext with the public key and then launch a brute-force attack to infer the encrypted query. Thus, people resort to the symmetric PE schemes [9], [10] to provide better search privacy at the price of significant usability deterioration in the case of GRQ (the strawman solution in Sect. IV-A). We may adopt a logarithmic-time B^+-tree based PE scheme proposed in [10]. However, for any ordered tree-based PE schemes, regardless of asymmetric or symmetric, the ordering information of the encrypted data will be directly disclosed. As a result, we can only achieve linear search when applying PE to GRQ at present.

Secure Keyword Search. Secure keyword search can be realized by either symmetric [14], [17], [18] or asymmetric [19], [20] cryptography. Curtmola *et al.* [14] proposed a searchable symmetric encryption (SSE) for single keyword search, and gave a formal security notion, i.e., *security against chosen-keyword attack* (CKA1) and a stronger and adaptive notion of CKA2. Sun *et al.* [18] proposed a UC-secure verifiable conjunctive keyword search scheme over dynamic encrypted cloud data in the malicious model. The verification cost only depends on search operation irrespective of the dataset size. In the public key scenario, Boneh *et al.* [19] presented a public key encryption with keyword search (PEKS) derived from identity-based encryption. In [20], the authors presented a verifiable attribute-based keyword search scheme, where only authorized users can obtain the intended search result.

Privacy-preserving Genomic Study. In the literature, privacy-preserving genomic data research has been extensively investigated to realize a variety of functionalities by the adoption of either secure multi-party computation (SMC) or secure outsourced computation techniques.

By using honey encryption, Huang *et al.* [21] proposed a secure outsourced genomic data storage scheme against the dictionary attack, where each decryption by the adversary even with an incorrect secret key will yield a valid-looking plaintext. Baldi *et al.* [5] presented a framework that aims for several important applications, such as paternity test, personalized medicine and genetic compatibility test by the generic secure two-party computation techniques. In [22], the authors presented an efficient secure edit distance approximation method, which is used to look for patients with similar genomic pattern (disease). Chen *et al.* [23] proposed a hybrid cloud-based scheme to delegate the partial read mapping task to the public cloud in a privacy-preserving manner.

There are few works towards secure and efficient solutions to the GRQ problem in the literature. The authors in [7] proposed a protocol that incorporates stream cipher, position scrambling and OPSE to store and retrieve the private raw genomic data. At the same time, the authors also expressed their concerns that this OPSE-based framework may not be secure for most practical applications [6], especially given a recent attack on OPSE [15]. Further, its practical efficiency is not satisfactory compared to ours.

III. Background

A. Biology Preliminaries

Genome represents the entirety of an organism's hereditary information, consisting of two long complementary polymer chains of four simple units called *nucleotides*, represented by letters A, C, G and T, which combine to form the double-stranded deoxyribonucleic acid (DNA) molecules in humans. There are approximately 3 billion nucleotides in 23 chromosomes of a human genome.

Single nucleotide polymorphisms (SNPs) are the most common form of DNA variation occurring when a single nucleotide (A, C, G, or T) in the genome differs between

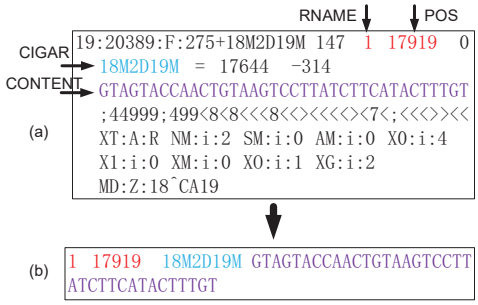

Fig. 1. Format illustration of a sample short read. (a) The original format in a SAM file; (b) Simplified read format

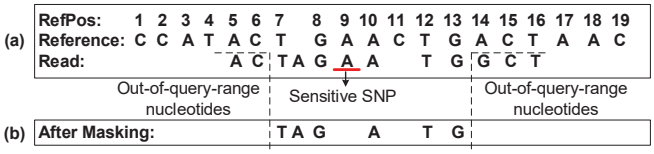

Fig. 2. (a) Short read alignment with the reference; (b) Content protection

members of the same species or paired chromosomes of an individual. SNPs often correspond to how humans develop diseases and respond to pathogens, chemicals, drugs, vaccines and other agents. Thus, they are usually adopted as fingerprints in a variety of genetic tests and genomic research.

Raw aligned genomic data for an individual is the aligned output of a DNA sequencer, comprised of millions of *short reads*, covering the entire genome of that person and subsequently aligned by using a reference genome. Each short read corresponds to a sequence of nucleotides within a DNA molecule. The position of a short read with respect to the reference genome is determined by the approximate match on the reference genome [6].

The raw alignment data can be stored in a sequence alignment/map (SAM) file, a BAM file (the binary version of SAM), or in a compressed CRAM file. Only SAM file is human readable albeit all the three formats contain the same alignment information [24] (We use SAM file for ease of illustration purpose only, but the proposed scheme is also applicable to BAM file). The format of a short read in a SAM file encompasses several data fields as shown in Fig. 1(a), three of which are considered privacy-sensitive.

The first is the position information PI=(RNAME,POS). RNAME is the name of the chromosome where this short read resides, and POS is the position of where this read maps to the reference chromosome. For example, the POS of the read in Fig. 2(a) relative to the reference is 5. As such, a *genome-wide range query* can be issued based on the positions PI of the intended short reads. For instance, with the range query $[(3, 100), (3, 150)]$, all the short reads, residing in between POS $=$ 100 and POS $=$ 150 of the reference chromosome 3, will be retrieved. The second sensitive field is the Concise Idiosyncratic Gapped Alignment Report (CIGAR) string, a sequence of nucleotide length and the associated operation, used to indicate which nucleotides align with the reference with letter `M`, are deleted from the reference with `D`, and are insertions not in the reference `I` (please refer to [24] for detailed exposition). Thus, the CIGAR string of the aligned read in Fig. 2(a) is `3M1I3M1D5M`. The CONTENT field consists of sensitive nucleotide information, e.g., SNPs. On the other hand, the rest of a short read are deemed non-private fields for a person. Hence, we skip the discussion of these fields hereafter and only focus on the simplified privacy-sensitive read format as shown in Fig. 1(b).

B. Secure GRQ Model

As shown in Fig. 3, a typical real-world secure GRQ system consists of four entities, *data contributor* (DC), *medical unit* (MU), *trusted authority* (TA), and *Genobank*. Specifically, DC can be a patient who submits his biospecimen to a certified institution for full genome sequencing. Then his raw genomic data are generated and uploaded to TA for data preprocessing before outsourced to the public Genobank. Subsequently, TA anonymizes the data, e.g., it will erase the identity information linked to a particular patient. The anonymized data will be used to generate a secure index and then encrypted, which are sent by TA to Genobank in the end.

MUs including biomedical researchers, physicians, pharmaceuticals, etc., may ask for a particular range of the alignment data of a patient for further genetic tests (processing GRQ for multiple patients may follow the same procedure as in the single patient situation). In this case, a GRQ request is sent to TA, which contains the identities of MU and the queried patient, intended genome range and query purpose (what kind of genetic test to be conducted later). It is worth noting that *the role of TA has been established in the plaintext GRQ before our design* and can be played by government agencies, like NIH, FDA, etc., to perform the authentication/authorization to MU and its query. Upon authentication, a private GRQ request is produced, and TA submits it to Genobank on behalf of MU.

Genobank runs the secure GRQ search algorithm with the private GRQ query over the stored secure indexes. The corresponding encrypted alignment data is then retrieved and sent back to TA. In the end, TA decrypts the data, masks the sensitive information based on the genomic privacy policy, and returns result to MU. As with previous works [7], [23], we assume that TA as the only authoritative party in the system is anticipated to have sufficient computation resources (e.g., dedicated server, private cloud, etc.), or can delegate the computation to other certified institutions. Furthermore, all the communication channels in the system are assumed to be secure. The discussion on the corresponding genomic data de-identification [25] and user authorization in the search phase [14], [20] is outside the scope of this paper.

C. Privacy Threats

We assume that DC is honest and not expected to be involved in the potential genetic tests [7]. TA is the key enabler entity in the system and acts as a private key manager/holder, standing between the user and Genobank. It conceals the

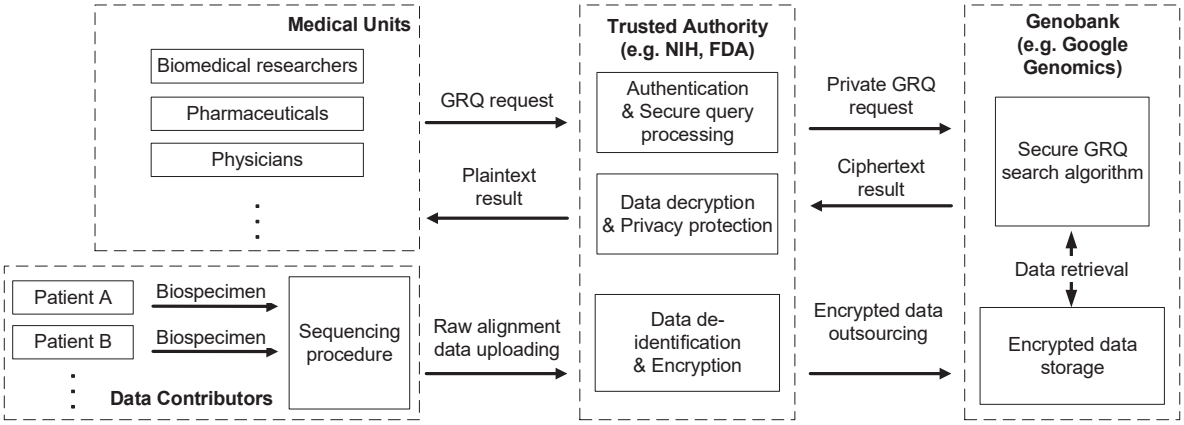

Fig. 3. Overview of secure GRQ system

– Real-world Experiment $\mathbf{Real}_{\mathcal{A}}(\lambda)$:
In the setup phase, the challenger runs $\mathsf{Setup}(1^{\lambda})$ to generate a key set K.
$\mathcal{A}$ sends a tuple $(\mathcal{IND}, \mathcal{F})$ to the challenger. He receives $\bar{\mathcal{F}} \leftarrow \mathsf{Enc}(\mathcal{F}, K)$ and secure index structure $(\mathcal{I\bar{N}D}_T, \{T_s\}) \leftarrow \mathsf{IndGen}(\mathcal{IND}, K)$.
In the search phase, for $i = 1, \ldots, q$,

- $\mathcal{A}$ chooses a keyword set $\mathcal{Q}_i$ and sends it to the challenger.
- The challenger returns an MSSE query $\bar{\mathcal{Q}}_i \leftarrow \mathsf{QueryGen}(\mathcal{Q}_i, K)$ to $\mathcal{A}$.

Finally, $\mathcal{A}$ outputs a bit b.

– Ideal-world Experiment $\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}(\lambda)$:
$\mathcal{A}$ chooses a tuple $(\mathcal{IND}, \mathcal{F})$. Given $\mathcal{L}_1(\mathcal{IND}, \mathcal{F})$, $\mathcal{S}$ outputs a tuple $(\mathcal{I\bar{N}D}_T, \{T_s\}, \bar{\mathcal{F}})$ and returns it to $\mathcal{A}$.
In the search phase, for $i = 1, \ldots, q$,

- $\mathcal{A}$ chooses a keyword set $\mathcal{Q}_i$.
- Given $\mathcal{L}_2(\mathcal{IND}, \mathcal{F}, \mathcal{Q}_i)$, $\mathcal{S}$ returns $\bar{\mathcal{Q}}_i$ to $\mathcal{A}$.

Finally, $\mathcal{A}$ outputs a bit b' in this experiment.

Fig. 4. Real-world and Ideal-world experiments

identity of MU so that the semi-honest Genobank cannot infer the underlying genetic test through such side information.

We assume that MU will not collude with Genobank to gain unauthorized access privileges. For MU, as shown in Fig. 2(b), we try to protect 1) *Out-of-query-range privacy*: We need to hide the *out-of-query-range* genomic information in the result short reads; 2) *Sensitive SNPs protection*: It requires the protection of sensitive SNP information *within* the queried range from MU, because these SNPs may happen to be indicators of other potential diseases irrelevant to the required genetic test [26].

Almost all the secure range search constructions [7], [9], [10], [13] leak *access pattern*, i.e., after searching in the dataset, the encrypted result is disclosed, and *search pattern*, i.e., whether the range was queried before, if the underlying query generation is deterministic. We do not aim to protect them due to the incurred expensive computation or/and communication overheads [27].

We define two stateful leakage functions $\mathcal{L}_1$ and $\mathcal{L}_2$ to precisely capture what is being revealed by the ciphertext and the query: 1) $\mathcal{L}_1(\mathcal{IND}, \mathcal{F})$. On input of the index $\mathcal{IND}$ and the genomic data file $\mathcal{F}$ containing millions of short reads $\mathcal{SR}$, this function outputs the number of short reads in $\mathcal{F}$, and the size of each short read $|\mathcal{SR}|$; 2) $\mathcal{L}_2(\mathcal{IND}, \mathcal{F}, \mathcal{Q})$. Besides $\mathcal{IND}$ and $\mathcal{F}$, this function also takes as input the query set $\mathcal{Q}$ searched in the past, and reveals *search* and *access pattern*. We first adapt the CKA2 security in [14] for an MSSE scheme that serves the core construction for our GRQ protocol in Fig. 5. Then we will reduce the security of the GRQ design to that of the MSSE (Sect. V). Specifically, we aim to ensure the confidentiality of a short read and its position information relative to the reference genome during the query phase.

Definition 1: (**CKA2 security for MSSE**) We consider two experiments in a real world and an ideal world respectively in Fig. 4, where $\mathcal{A}$ is a stateful adversary, $\mathcal{S}$ is a stateful simulator, and $\mathcal{L}_1$ and $\mathcal{L}_2$ are stateful leakage functions described above.

We say that the proposed MSSE scheme is CKA2 secure if for all probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, there exists a PPT simulator $\mathcal{S}$ such that

$$|Pr[\mathbf{Real}_{\mathcal{A}}(\lambda) = 1] - Pr[\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}(\lambda) = 1]| \leq \mathrm{neg}(\lambda).$$

IV. Secure GRQ Construction

In this section, we first see how a conventional symmetric PE-based strawman solution fails the design goals (Sect. IV-A). Then we present our efficient and scalable secure GRQ scheme (Sect. IV-B) based on a novel hierarchical index structure and improve the search efficiency by filtering out the irrelevant query terms in a pre-search stage (Sect. IV-C).

A. Strawman Solution

PE has been adopted widely to achieve secure range query on outsourced encrypted data [9], [10], where only the ciphertext within the range of interest can be decrypted. Hence, seemingly we can design a secure GRQ scheme by trivially using symmetric PE for better *predicate privacy* [9]. However, when PE is applied to an enormous volume of data, such as human genome, efficiency and scalability become the primary concerns. For example, by using symmetric inner-product predicate-only encryption [9], [10], the encrypted index size for a single short read is proportional to the position domain size $N = 3 \times 10^9$, which leads to a significant ciphertext size expansion. Besides, the computation cost for encryption, token generation, and single read query operation are all proportional

Setup
$K \leftarrow \mathsf{Setup}(1^\lambda)$. On input of a security parameter λ, this probabilistic algorithm outputs a secret key set $K = (k_1, k_2)$.

Data Upload
$\bar{\mathcal{F}} \leftarrow \mathsf{Enc}(\mathcal{F}, K)$. To encrypt a short read $\mathcal{SR}_i$ in $\mathcal{F}$, TA generates a random nonce r_i, and then acquires the encrypted read $\bar{\mathcal{SR}}_i$ from the first l_i bits of $\mathcal{SR}_i^b \oplus h_{k_2}(r_i)$. $\oplus$ denotes the bit-wise XOR operation. In the end, all the short read ciphertexts $\mathcal{C}_i = \{\bar{\mathcal{SR}}_i, r_i\}$ for this patient are stored in a secure genomic data file $\bar{\mathcal{F}}$.
$(\mathcal{I\bar{N}D}_T, \{T_s\}) \leftarrow \mathsf{IndGen}(\mathcal{IND}, K)$. Let $\mathcal{IND} = \{\mathsf{PI}_i\}_{1 \le i \le n_1}$ be an *ordered* index set for all the short reads in $\mathcal{F}$. Then TA first generates an obfuscated index set $\mathcal{I\bar{N}D} = \{h(i)\}$ with $h(i) = h_{k_1}(\mathsf{PI}_i)$. Next, $\mathcal{IND}$ is sorted to an ordered set as per the value of $h(i)$, which is further split into d partitions with equal length q such that $dq = n_1$. In addition, for each partition $s = 1, ..., d$, an ordered index table T_s is produced with q tuples in the form of $\{h((s-1)q + z), off_{(s-1)q+z}\}$, $1 \le z \le q$. $off_{(s-1)q+z}$ is the offset used to pinpoint the corresponding short read ciphertext. On the other hand, in each partition s, TA uses the highest ordering value $\{h(sq), pt_s\}$ to represent this partition range. pt_s points to the index table T_s. As such all the tuples for the d partitions constitutes a sorted range index $\mathcal{I\bar{N}D}_T$ for d index tables. Finally, TA uploads $\bar{\mathcal{F}}$ along with the secure index structure $\mathcal{I\bar{N}D}_T$ and $\{T_s\}_{1 \le s \le d}$ to Genobank.

Secure GRQ Search
$\bar{\mathcal{Q}} \leftarrow \mathsf{QueryGen}(\mathcal{Q}, K)$. On receiving a GRQ request with the range $[L, U]$ from an authorized MU, TA re-defines the query range $[L, U]$ to $[L', U']$ where $L' = L - a$, $U' = U + a$ and a is a random number greater than the size of the longest short read. Then he generates a query set, $\mathcal{Q} = \{\mathsf{PI}_i\}$ for $L' \le \mathsf{PI}_i \le U'$ and further the scrambled (encrypted) query set $\bar{\mathcal{Q}}$ using pseudorandom function h_{k_1} and a random permutation. TA submits $\bar{\mathcal{Q}}$ to Genobank on behalf of MU.
$L_{\bar{\mathcal{Q}}} \leftarrow \mathsf{Search}(\bar{\mathcal{Q}}, \mathcal{I\bar{N}D}_T, \{T_s\}, \bar{\mathcal{F}})$. Upon receipt of $\bar{\mathcal{Q}}$, Genobank performs a binary search with each term in $\bar{\mathcal{Q}}$ on the sorted range index $\mathcal{I\bar{N}D}_T$ which leads the query process to the relevant second-level index tables. Subsequently, Genobank conducts the binary search again on each related table T_s to retrieve the intended short read ciphertexts from $\bar{\mathcal{F}}$, puts them in the result list $L_{\bar{\mathcal{Q}}}$.

Data Download
$\mathcal{SR} \leftarrow \mathsf{Dec}(\mathcal{C}, K)$. TA decrypts each encrypted short read $\mathcal{C} = \{\bar{\mathcal{SR}}, r\}$ in $L_{\bar{\mathcal{Q}}}$ by $\bar{\mathcal{SR}} \oplus h_{k_2}(r)$ and checks whether PI of each plaintext $\mathcal{SR}$ falls in the range of interest $[L', U']$. TA also deletes the out-of-query-range nucleotides and the within-range sensitive SNPs in the $\mathsf{CONTENT}$ field as per the range $[L, U]$, and modifies the CIGAR string accordingly. Finally, TA returns the sanitized search result to MU.

Fig. 5. Proposed secure GRQ construction

to N as well. Note that the dominant computation in the query phase is the composite-order pairing operation. Therefore, the service provider is not likely to favor such expensive computation approach. When $\Theta(N)$ pairing operations are required to search merely one short read, the scheme becomes unrealistic in a pay-as-you-go manner, even if the computation can be delegated to a more powerful cloud.

Last but not the least, the best search complexity we could achieve for this strawman solution is linear search so as to protect the query privacy, since using any sorted tree structure, e.g. B^+-tree [10], will outright leak the ordering of ciphertexts in the tree. Then the adversary can estimate the queried range relative to the reference genome, which may disclose the underlying genetic test.

Observations. We observe that the straightforward order-comparison methods used for the generic range query on encrypted data [8], [13] or logarithmic-time search over order-revealed sorted data structure [10] will not apply to the range-sensitive GRQ service. To break such linkability, it requires us to look for a novel alternative approach, in contrast to the "off-the-shelf" methods, to protect the range information while realizing efficient query.

B. Our Construction

We propose a secure GRQ solution using a lightweight MSSE construction in Fig. 5. It utilizes a scalable and efficient secure index structure, which offers not only genomic privacy preservation but also a logarithmic-time search complexity.

Suppose there are total n nucleotides in the reference genome and n_1 short reads in a patient's genomic data file $\mathcal{F}$. Thus, the universal position domain $\mathcal{P}$ contains n distinct PI. Let a short read $\mathcal{SR}_i = \{\mathsf{PI}_i, \mathsf{CIGAR}_i, \mathsf{CONTENT}_i\}$, $i = 1, ..., n_1$. Let $\mathcal{SR}_i^b$ be the l_i-bit long binary form of $\mathcal{SR}_i$.

Definition 2: (**Ordered Set**) We say $S = (s_1, s_2, ...s_N)$ of size N is an *ordered set* if its numerical terms are sorted by an ascending order such that $s_1 < s_2 < ... < s_N$.

Definition 3: (**Scrambled Set**) For an *ordered set* $S = (s_1, s_2, ...s_N)$, applying a pseudorandom function h_k with secret key k and a random permutation σ to S yields a *scrambled (encrypted) set* $\bar{S} = \{h_k(s_{\sigma(1)}), h_k(s_{\sigma(2)}), ..., h_k(s_{\sigma(N)})\}$.

1) Efficient Data Protection: Typically, storing the raw alignment data for a patient is significantly space-consuming, e.g., even with the compressed data format CRAM, the file size can be easily over 20 GB. The underlying data encryption is expected to be efficient in both storage and computation. As shown in Fig. 5, the short read is XORed with a pseudorandom bit string produced from a random nonce. Subsequently, a collection of the encrypted short reads along with the corresponding random numbers and other encrypted auxiliary information in the original SAM/BAM file are all stored in one secure genomic data file $\bar{\mathcal{F}}$. This design is consistent with the real-world situation, where a single SAM/BAM file is used to store all the raw alignment data of a person. Another key observation behind this design is that storing each encrypted short read as an individual file can have an adverse impact on the system because hundreds of millions of small files present in the file system can degrade the cloud performance [12]. Moreover, the query process depends on the file storage lookup, which is often not specialized for range query problem.

It is worth noting that TA does not directly send the queried short reads back to MU after decryption. Instead, he re-writes part of the result to prevent MU learning additional information beyond the requested. To protect the *out-of-query-range privacy* and *sensitive SNPs* within the range

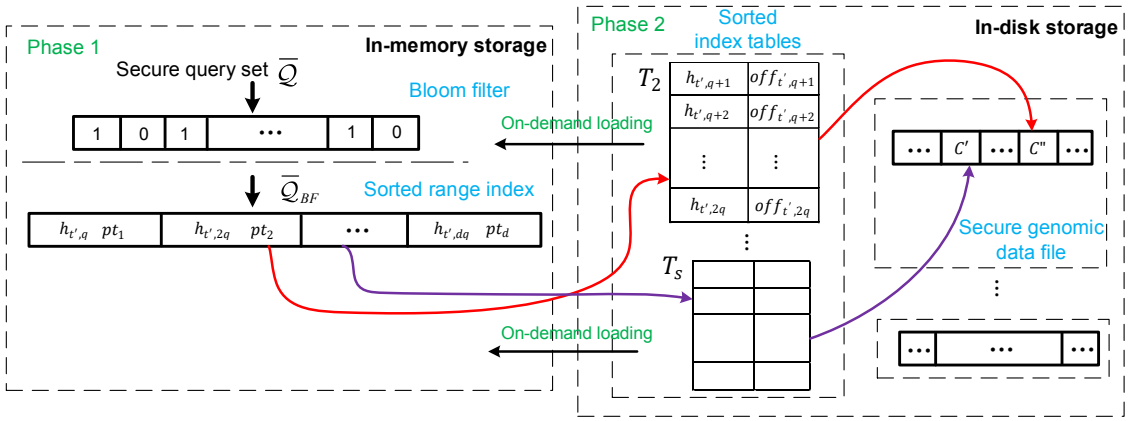

Fig. 6. Hierarchical GRQ-orientated index structure with a two-phase query process

Algorithm 1 Two-phase Query Algorithm

Input: Preloaded sorted range index $\mathcal{I}\bar{\mathcal{N}}\mathcal{D}_T$ for d partitions with equal length q, in-disk table indexes $\{T_s\}_{1\leq s\leq d}$, secure genomic data file $\bar{\mathcal{F}}$, and scrambled GRQ query set $\bar{\mathcal{Q}} = \{\bar{h}_{k_1}(\mathsf{PI}_i)\}$ for $L' \leq \mathsf{PI}_i \leq U'$.
Output: GRQ query result $L_{\bar{\mathcal{Q}}}$

1: Let $L_{\bar{\mathcal{Q}}} = \emptyset$ ▷ Phase 1 query
2: **for** $j = 1 \rightarrow U' - L' + 1$ **do**
3: Perform a binary search with each element in $\bar{\mathcal{Q}}$ over $\mathcal{I}\bar{\mathcal{N}}\mathcal{D}_T$ in memory
4: Pinpoint the second-level index T_s in disk with the corresponding pointer pt_s in the partition range
5: Load T_s into memory
▷ Phase 2 query
6: Perform a binary search with $h_{k_1}(\mathsf{PI}_i)$ over T_s in memory
7: Find out the file offset in T_s of the intended encrypted short read and use it to retrieve the short read from $\bar{\mathcal{F}}$ in disk
8: Put the result in $L_{\bar{\mathcal{Q}}}$
9: **end for**
return $L_{\bar{\mathcal{Q}}}$

(Sect. III-C), TA modifies the CONTENT and the associated CIGAR string of the corresponding short read. For example, in Fig. 2(b), the CIGAR string is changed from `3M1I3M1D5M` to `1M1I1M1D1M1D2M` in accordance with the privacy policy.

2) Secure GRQ Request Processing: Upon receipt of the original GRQ request from DC, TA needs to transform the query to a secure version. Specifically, TA generates a random number a, greater than the number of nucleotides in the longest short read in the SAM file. He then derives a new query range $[L', U']$ from the original $[L, U]$ as in Fig. 5 in the sense that a short read with its position in $[L', L-1]$ may also contain the nucleotides in $[L, U]$. To preserve the symmetry property, TA also modifies U to $U + a$. Then the query set size is $U' - L' + 1$. In the case of the query approaching the end of the genome, TA renders a an appropriate value that may be smaller than the maximum short read length. Finally, the queried range can be represented by an *ordered set* $\mathcal{Q}$ and later turned into a *scrambled set* $\bar{\mathcal{Q}}$ as the secure query set.

3) Scalable GRQ-oriented Index Structure in Cloud: In order to enable large-scale privacy-preserving genomic research, it is imperative to store data in an efficient and scalable data structure. However, the current *de facto* genomic data organization is SAM/BAM file specification [24]. The binary BAM file is more data-processing friendly, but also has been criticized for its lack of scalability in a multiprocessing environment due to the use of a centralized header [11].

Our GRQ-oriented index structure involves a two-level design, focusing on efficiently retrieving the intended short reads from the secure genomic data file. As shown in Fig. 6, a primary index $\mathcal{I}\bar{\mathcal{N}}\mathcal{D}_T$ is devised and resides in the memory as the initial coarse-grained search phase. The benefit of this design is that it is not necessary to load the large index set $\mathcal{I}\bar{\mathcal{N}}\mathcal{D}$ for a patient's entire raw alignment data into memory. Instead, a succinct index representation $\mathcal{I}\bar{\mathcal{N}}\mathcal{D}_T$ is used to quickly pinpoint the relevant index range, which is organized as a secondary index table $\{T_s\}_{1\leq s\leq d}$, kept in the disk storage. To avoid the small-file problem in the cloud [12], these second-level index tables can be stored in a file and fetched by the pointers in $\mathcal{I}\bar{\mathcal{N}}\mathcal{D}_T$. T_s stores the exact mapping from each secure index to the file offset of the encrypted short read in $\bar{\mathcal{F}}$. Due to the saved space from $\mathcal{I}\bar{\mathcal{N}}\mathcal{D}_T$, it is possible to load larger index tables into memory on demand after *Phase 1* search over $\mathcal{I}\bar{\mathcal{N}}\mathcal{D}_T$ so as to enable fine-grained *Phase 2* search. As our proposed index structure are sorted by the secure index values, any efficient logarithmic-time search algorithm can be applied here. It is also worth noting that with the in-memory primary index, it is possible to enable large-scale concurrent lookups compared to the file-based indexing scheme currently used by BAM file. The pseudo code of the efficient two-phase query algorithm is shown in Algorithm 1.

Observations. The computation complexity of the proposed secure GRQ search is $\Theta(m\lg(n_1))$. The secure query set size m is possible to be as large as the size of the position domain $\mathcal{P}$, which means a considerable number of terms in $\bar{\mathcal{Q}}$ will not have a match in the following search process, thereby wasting the computation resources in practice. Should we pre-screen the query term before throwing it to the search process, a significant amount of time may be saved in a long run.

C. Improving Search Efficiency

Based on the above observations, we propose a pre-search stage to examine the existence of the query term to boost

search efficiency as shown in Fig. 6. More precisely, we adopt Bloom filter [28] on top of the existing secure index structure to provide efficient membership test. The Bloom filter is generated from the obfuscated index set $\mathcal{I}\bar{\mathcal{N}}\mathcal{D}$ with controlled false positive rate (FPR). By executing the membership test for terms in $\bar{\mathcal{Q}}$ prior to the two-phase query process, we can narrow $\bar{\mathcal{Q}}$ down to a likely smaller query set $\bar{\mathcal{Q}}_{\mathrm{BF}}$ taken as the input of the subsequent secure search algorithm.

Discussion. Bloom filter will only introduce false positive but not false negative. This is desirable since no terms indeed belonging to the result will fail the membership test. Then we always have a complete search result. Furthermore, the false positive will not appear in the final GRQ result because such error can be rectified in the following exact-match search process. Notably, the membership test does not disclose the location of the short read in $\bar{\mathcal{F}}$. Linearly searching the entire file is prohibitively expensive. We still need to rely on our efficient search algorithm over the proposed genomic index structure to retrieve the short reads of interest.

V. Security Analysis

In this section, we show that our proposed secure GRQ scheme will achieve the defined security goals.

Theorem 1: (Privacy against MU) The proposed secure GRQ scheme satisfies the out-of-query-range privacy and sensitive SNPs protection requirement for MU.

Proof: (sketch) To prevent MU from acquiring the out-of-request information from the query result, TA re-writes the corresponding fields of the short read results before returning them to MU. More precisely, he modifies the CONTENT field to delete the irrelevant sensitive SNPs and truncate the short read as per the query range $[L, U]$. TA also produces a new CIGAR string accordingly. In the end, the *out-of-query-range privacy* and *sensitive SNPs* can be well protected against MU. ■

The centerpiece of the proposed secure GRQ design is an MSSE construction in the sense that we may deem position information PI of each short read a unique keyword. Intuitively, the confidentiality of a short read and its position information are ensured if the underlying MSSE scheme is CKA2 secure. Therefore, the security offered by our scheme reduces to that of the MSSE protocol.

Lemma 1: (CKA2 security for MSSE) The proposed MSSE scheme achieves CKA2 security [14], in the random oracle model defined in Def. 1.

Proof: Let $\mathcal{A}$ and $\mathcal{S}$ be an adversary and a simulator in $\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}(\lambda)$ in Def. 1, respectively. Assume that each $\mathcal{SR}$ in $\mathcal{F}$ has a unique PI. Given the leakage function $\mathcal{L}_1(\mathcal{IND}, \mathcal{F})$, $\mathcal{S}$ outputs $(\bar{\mathcal{F}}', \mathcal{I}\bar{\mathcal{N}}\mathcal{D}'_T, \{T'_s\})$ as follows. It simulates each encrypted short read $\mathcal{C}'_i = \mathsf{Enc}(k_2, 0^{|\mathcal{SR}_i|})$ in $\bar{\mathcal{F}}'$ for $i = 1, ..., n_1$, where k_2 is randomly selected for the CPA-secure encryption algorithm Enc, and $|\mathcal{SR}_i|$ is revealed by $\mathcal{L}_1$. To simulate the secure index structure $\mathcal{I}\bar{\mathcal{N}}\mathcal{D}'_T$ and $\{T'_s\}$, $\mathcal{S}$ first sets $h'(i)$ for each $\mathcal{SR}_i$ in $\mathcal{F}$ a random number, and then generates $\mathcal{I}\bar{\mathcal{N}}\mathcal{D}'_T$ and $\{T'_s\}$ accordingly. The Bloom filter can also be simulated by using random numbers instead of cryptographic hashing.

Adversary $\mathcal{A}$ can make polynomial number of queries by picking a keyword set $\mathcal{Q}$ of t continuous position information PI for $t \geq 2$. The leakage function $\mathcal{L}_2(\mathcal{IND}, \mathcal{F}, \mathcal{Q})$ discloses $L_{\bar{\mathcal{Q}}}$ to $\mathcal{S}$. Given this, $\mathcal{S}$ can simulate $\bar{\mathcal{Q}}'$ by including $h'(i)$ for the short reads in $L_{\bar{\mathcal{Q}}}$ and assigning the remaining in $\bar{\mathcal{Q}}$ random numbers if $t > |L_{\bar{\mathcal{Q}}}|$.

Adversary $\mathcal{A}$ cannot distinguish $\bar{\mathcal{F}}'$ from $\bar{\mathcal{F}}$ in experiment $\mathbf{Real}_{\mathcal{A}}(\lambda)$ since Enc is CPA-secure. Due to the pseudorandom function h_k and cryptographic hash function, $\mathcal{A}$ cannot distinguish $(\mathcal{I}\bar{\mathcal{N}}\mathcal{D}'_T, \{T'_s\})$ and $(\mathcal{I}\bar{\mathcal{N}}\mathcal{D}_T, \{T_s\})$, and the Bloom filters. Likewise, $\mathcal{A}$ cannot discern $\bar{\mathcal{Q}}'$ and $\bar{\mathcal{Q}}$. Thus, $\mathcal{A}$ cannot distinguish $\mathbf{Ideal}_{\mathcal{A},\mathcal{S}}(\lambda)$ and $\mathbf{Real}_{\mathcal{A}}(\lambda)$. ■

Theorem 2: (Security for the GRQ scheme) The proposed secure GRQ scheme enjoys the same security guarantees as the underlying MSSE construction, i.e., Biobank is not able to learn any information about the content of the short reads and position information in the query and secure index during the search phase.

Proof: (sketch) This is inherited from the proof of Lemma 1, by which our secure GRQ scheme achieves the CKA2 security for MSSE. In other words, the adversary cannot deduce the content of an encrypted genomic file $\bar{\mathcal{F}}$, and the position information of the short reads from the secure index and query in the random oracle model. ■

Impact of Access Pattern Leakage. Indeed, to provide a practical GRQ solution, we need to accept some measure of leakage. Thus, one of our design goals is to achieve an acceptable balance between performance and leakage. Given that access pattern is disclosed after the query, we find that limited partial ordering information will be revealed to the adversary, which also applies to almost all the secure range query constructions, regardless of the underlying primitives, i.e. PE [9], [10] (the strawman scheme), OPSE [7], etc. Specifically, the ascending or descending data order will be inevitably exposed if a motivated adversary observes sufficient query results, but still he cannot determine which order is correct. For instance, consider 3-record dataset (a, b, c). Having observed two range query results (b, a) and (c, b), adversary can figure out the possible data orders, either (a, b, c) or (c, b, a). We should note that, albeit it is trivial to perform such attack on a relatively small dataset, it is unlikely that the query results would spread over the whole genome, instead of sporadic genetic hotspots in light of our very constrained knowledge on human genome at present. Compared to existing solutions, our secure GRQ scheme, as an initial attempt, enjoys logarithmic-time query process, and is considerably scalable and suitable for real-world cloud environment, with equivalent or better security guarantees. We may adopt "heavy" cryptographic tools on top of our protocol, such as Oblivious RAM [29], for mitigation but at the price of sacrificing the practical usability and efficiency [27].

TABLE I
PERFORMANCE OF THE PROPOSED SECURE GRQ SCHEME WITHOUT PRE-SEARCH STAGE

d	q	Index (MB)		Indexing time (s)	Loading time (s)	Query time (s) with different range size						
		$\mathcal{I}\bar{\mathcal{N}}\mathcal{D}_T$	T_s			100	500	1,000	5,000	10,000	50,000	100,000
21,845	14,388	1	0.66	546	0.21	0.91	3.98	7.87	38.6	75.46	385.33	761.78
16,384	19,184	0.75	0.88	778	0.18	1.44	7.99	11.86	59.41	105.47	583.87	1,017.5
10,922	28,776	0.5	1.32	790	0.14	1.96	9.94	17.16	83.65	161.12	803.78	1,568.62
5,461	57,553	0.25	2.63	758	0.11	3.71	16.1	32.09	165.88	318.1	1,636.79	3,223.33
2,184	143,884	0.1	6.59	817	0.09	8.57	39.8	79.66	401.47	805.16	3,895.04	9,071.23
1,092	287,769	0.05	13.17	1,099	0.07	16.62	79.16	159.48	791.59	1,656.91	7,947.97	18,046.59

VI. PERFORMANCE EVALUATION

In this section, we implement the proposed secure GRQ scheme using real human raw alignment data in an approximately 32 GB CRAM file [30]. After decompression, the BAM file is about 44 GB, which contains more than 300 million short reads with the average read length of 100 nucleotides. We use JAVA and shell scripts on a Linux server with a 3.1 GHz AMD FX 8120 processor and 32 GB DDR3 memory. The server runs on a WD100ZFAEX hard disk with 1 TB storage and 64 MB disk cache. We use SHA256 to construct the Bloom filter in the pre-search stage. The experimental result is an average of 10 trials.

A. Storage Overhead

1) Ciphertext Expansion: We measure the storage overhead on Genobank for the encrypted raw alignment data of a patient and its secure index structure. Note that the bitwise XOR encryption in our scheme introduces no ciphertext expansion even for the huge-sized genomic data. In practice, the encryption could be instantiated by CRT[AES]. Besides, an additional 1.26 GB storage burden comes from the 4-byte nonce for each short read in the dataset.

2) Scalability: The strawman construction will consume prohibitive space, i.e. 700 GB or so for each short read index since the ciphertext size is linear to the whole genome length, thereby crippling its usability in practice. For real-world implementation, it is straightforward to load the entire secure index $\mathcal{I}\bar{\mathcal{N}}\mathcal{D}$ into cloud memory, which requires roughly 14 GB with our scheme. It is significantly smaller than the strawman solution but still not practical for processing multiple GRQ queries concurrently even with the cloud. On the contrary, using our hierarchical secure index design, the cloud memory consumption for the primary index $\mathcal{I}\bar{\mathcal{N}}\mathcal{D}_T$ is orders of magnitude smaller than both the strawman and heuristic implementation. As shown in Table I, suppose that storing $\mathcal{I}\bar{\mathcal{N}}\mathcal{D}_T$ in memory for each patient costs 1 MB, and we can use up to all the 32 GB server memory, which allows hosting more than 300 thousand patients' primary indexes in the memory simultaneously and demonstrates the significant scalability of our scheme. The total size of the second-level table indexes $\{T_s\}_{1\leq s\leq d}$ is comparable to the secure index $\mathcal{I}\bar{\mathcal{N}}\mathcal{D}$ in the heuristic implementation but stored in the disk and loaded into memory on demand. Moreover, we can speed up the query process by spending an additional storage space for the Bloom filter (e.g. 200 MB with FPR 6%).

B. Time Efficiency

1) Encryption and Decryption: It takes less than 4 minutes for TA to encrypt more than 300 million short reads. This overhead is one-time and can be further amortized by parallelization. Decryption will be much faster since the query result usually is a much smaller subset of the whole genome.

2) Index Generation: This procedure also imposes a one-time cost to TA. The computation overhead varies with the patient's genomic data size and index construction parameter d. We use Linux shell script to perform several data preprocessing tasks, including extracting positions of short reads in the BAM file, generating their secure representation as well as sorting them in the index. It takes 36 minutes to preprocess the raw genomic data. As shown in Table I, for larger partition number d, less time is required to generate the secure index. This variation is due to the use of object serializer in our Java code. The larger d is, the smaller each second-level in-disk index table is, thus requiring less memory maneuvering in JVM and resulting in the gain in processing speed. For the implementation of our scheme that does not make use of the object serializer function, we expect the performance penalty of a smaller d to diminish. On the other hand, generating Bloom filter in the pre-search stage needs nearly constant time, roughly 20 minutes, for different FPR, because the number of short reads to index for a person is constant.

3) Query Efficiency: Table I shows that query is more efficient with an increased d value. Larger d implies bigger first-level index, and thus a longer initial loading time. On the other hand, since the loading time for the primary index is less than 1 second, we believe that the load time penalty is not significant. After this one-time loading, $\mathcal{I}\bar{\mathcal{N}}\mathcal{D}_T$ resides in memory to facilitate expediting the query process. Moreover, a larger d triggers a more fine-grained first-level search. As a result, the on-demand secondary index loading is much faster due to its reduced size. We randomly select a GRQ query with different range size. The query time in Table I shows a linearity with the increased range size. We can see that our GRQ scheme is still efficient enough for practical use even with a large query range. In contrast, query time for the

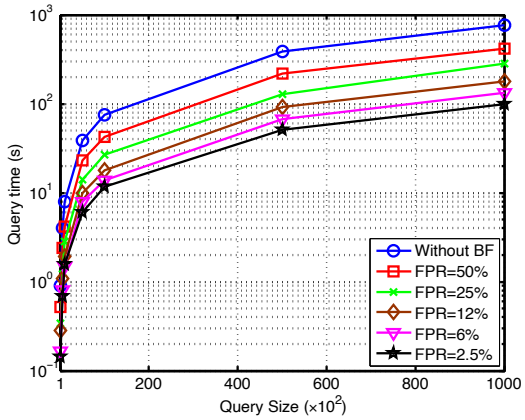

Fig. 7. Query time for different search strategies with query size 100, 500, 1,000, 5,000, 10,000, 50,000, 100,000. The pre-search stage is operated on the baseline search with $d = 21,845$.

PE-based strawman solution is prohibitively expensive. Using the state-of-the-art benchmark for the costly composite-order pairing operation [31], it will take more than one year to search only one short read index. Notably, the OPSE-based GRQ scheme [7] reports a 4.5 second query time with a small 100 query range size. Lastly, we can also use Bloom filter in a pre-search stage to further ameliorate query efficiency. As shown in Fig. 7, query time is significantly reduced by using the Bloom filter with a small FPR.

VII. Conclusion

Genome-wide range query is one of the fundamental and critical components in genomic research, medical and healthcare services. In this paper, we are among the first to propose a scalable, privacy-preserving GRQ scheme in the cloud environment, featuring an efficient hierarchical index structure. By presenting a novel MSSE-based secure range query scheme, our solution also enjoys storage and computation efficiency without sacrificing security guarantees. The implementation with real human raw alignment data shows its superiority over the existing solutions.

Acknowledgment

This work was supported in part by the US National Science Foundation under grants CNS-1446478, CNS-1405747, and CNS-1217889.

References

[1] National Human Genome Research Institute, http://www.genome.gov/sequencingcosts/, accessed: 10-2015.
[2] L. J. Young, "Genomic data growing faster than twitter and youtube," http://spectrum.ieee.org/tech-talk/biomedical/diagnostics/the-human-os-is-at-the-top-of-big-data, accessed: 10-2015.
[3] S. S. Shringarpure and C. D. Bustamante, "Privacy risks from genomic data-sharing beacons," *AJHG*, 2015.
[4] FDA, "Biomarker qualification program," http://www.fda.gov/Drugs/DevelopmentApprovalProcess/DrugDevelopmentToolsQualificationProgram/ucm284076.htm, accessed: 10-2015.
[5] P. Baldi, R. Baronio, E. De Cristofaro, P. Gasti, and G. Tsudik, "Countering GATTACA: Efficient and secure testing of fully-sequenced human genomes," in *Proc. of ACM CCS'11*, pp. 691–702.
[6] M. Naveed, E. Ayday, E. W. Clayton, J. Fellay, C. A. Gunter, J.-P. Hubaux, B. A. Malin, and X. Wang, "Privacy in the genomic era," *ACM CSUR*, vol. 48, no. 1, p. 6, 2015.
[7] E. Ayday, J. L. Raisaro, U. Hengartner, A. Molyneaux, and J.-P. Hubaux, "Privacy-preserving processing of raw genomic data," in *DPM'14*, pp. 133–147.
[8] A. Boldyreva, N. Chenette, Y. Lee, and A. O'Neill, "Order-preserving symmetric encryption," in *EUROCRYPT'09*, pp. 224–241.
[9] E. Shen, E. Shi, and B. Waters, "Predicate privacy in encryption systems," in *Proc. of TC'09*, pp. 457–473.
[10] Y. Lu, "Privacy-preserving logarithmic-time search on encrypted data in cloud." in *Proc. of NDSS'12*.
[11] M. Niemenmaa, A. Kallio, A. Schumacher, P. Klemelä, E. Korpelainen, and K. Heljanko, "Hadoop-bam: directly manipulating next generation sequencing data in the cloud," *Bioinformatics*, vol. 28, no. 6, pp. 876–877, 2012.
[12] T. White, "The small files problem," http://blog.cloudera.com/blog/2009/02/the-small-files-problem/, accessed: 11-2015.
[13] D. Boneh, K. Lewi, M. Raykova, A. Sahai, M. Zhandry, and J. Zimmerman, "Semantically secure order-revealing encryption: Multi-input functional encryption without obfuscation," in *EUROCRYPT'15*, pp. 563–594.
[14] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proc. of ACM CCS'06*, pp. 79–88.
[15] M. Naveed, S. Kamara, and C. V. Wright, "Inference attacks on property-preserving encrypted databases," in *Proc. of ACM CCS'15*, pp. 644–655.
[16] E. Shi, J. Bethencourt, T. H. Chan, D. Song, and A. Perrig, "Multi-dimensional range query over encrypted data," in *Proc. of IEEE S&P'07*, pp. 350–364.
[17] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li, "Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking," *IEEE TPDS*, vol. 25, no. 11, pp. 3025–3035, 2014.
[18] W. Sun, X. Liu, W. Lou, Y. T. Hou, and H. Li, "Catch you if you lie to me: Efficient verifiable conjunctive keyword search over large dynamic encrypted cloud data," in *Proc. of IEEE INFOCOM'15*, pp. 2110–2118.
[19] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *EUROCRYPT'04*, pp. 506–522.
[20] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li, "Protecting your right: Verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," *IEEE TPDS*, vol. 27, no. 4, pp. 1187–1198, 2016.
[21] Z. Huang, E. Ayday, J. Fellay, J.-P. Hubaux, and A. Juels, "Genoguard: Protecting genomic data against brute-force attacks," in *Proc. of IEEE S&P'15*, pp. 447–462.
[22] X. S. Wang, Y. Huang, Y. Zhao, H. Tang, X. Wang, and D. Bu, "Efficient genome-wide, privacy-preserving similar patient query based on private edit distance," in *Proc. of ACM CCS'15*, pp. 492–503.
[23] Y. Chen, B. Peng, X. Wang, and H. Tang, "Large-scale privacy-preserving mapping of human genomic sequences on hybrid clouds," in *Proc. of NDSS'12*.
[24] SAMTools, "Sequcence alignment/map format specification," https://samtools.github.io/hts-specs/SAMv1.pdf, accessed: 09-2015.
[25] D. M. Roden, J. M. Pulley, M. A. Basford, G. R. Bernard, E. W. Clayton, J. R. Balser, and D. R. Masys, "Development of a large-scale de-identified DNA biobank to enable personalized medicine," *Clinical Pharmacology & Therapeutics*, vol. 84, no. 3, pp. 362–369, 2008.
[26] Eupedia, http://www.eupedia.com/genetics/medical_dna_test.shtml, accessed: 10-2015.
[27] M. Naveed, "The fallacy of composition of oblivious ram and searchable encryption," Tech. rep., Cryptology ePrint Archive, Report 2015/668, Tech. Rep., 2015.
[28] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.
[29] O. Goldreich, "Towards a theory of software protection and simulation by oblivious rams," in *Proc. of TC'87*, pp. 182–194.
[30] 1000 Genome Project, ftp://ftp-trace.ncbi.nih.gov/1000genomes/ftp/data/HG00097/alignment/, accessed: 10-2015.
[31] Y. Zhang, C. J. Xue, D. S. Wong, N. Mamoulis, and S. M. Yiu, "Acceleration of composite order bilinear pairing on graphics hardware," in *Proc. of ICICS'15*, pp. 341–348.