Open access • Proceedings Article • DOI:10.1145/2517840.2517849

Secure genomic testing with size- and position-hiding private substring matching

Emiliano De Cristofaro, Sky Faber, Gene Tsudik

Institutions: PARC, University of California, Irvine

Published on: 04 Nov 2013 - Workshop on Privacy in the Electronic Society

Topics: Substring and Genomics

Secure Genomic Testing with Size- and Position-Hiding Private Substring Matching

Emiliano De Cristofaro[1] Sky Faber[2] Gene Tsudik[2]

[1] Palo Alto Research Center
Palo Alto, CA
me@emilianodc.com

[2] University of California, Irvine
Irvine, CA
{fabers,gene.tsudik}@uci.edu

ABSTRACT

Recent progress in genomics and bioinformatics is bringing complete and on-demand sequencing of human (and other) genomes closer and closer to reality. Despite exciting new opportunities, affordable and ubiquitous genome sequencing prompts some serious privacy and ethical concerns, owing to extreme sensitivity and uniqueness of genomic information. At the same time, new medical applications, such as personalized medicine, require testing genomes for specific markers that themselves represent sensitive (e.g., proprietary) material. This paper focuses on privacy challenges posed by such genetic tests. It presents a secure and efficient protocol called: Size- and Position-Hiding Private Substring Matching (SPH-PSM). This protocol allows two parties – one with a digitized genome and the other with a set of DNA markers – to conduct a test, such that the result is only learned by the former, and no other information is learned by either party. In particular, the genome owner does not even learn the size or the position of the markers, which makes SPH-PSM the first of its kind. Finally, we report on a prototype of the proposed technique which attests to its practicality.

Categories and Subject Descriptors: E.3 [Data Encryption]: Secure Multi-party Computation

General Terms: Security.

Keywords: Privacy, DNA, Cryptographic Protocols.

1. INTRODUCTION

Whole Genome Sequencing (WGS) is a revolutionary technology that determines the complete genetic blueprint of any organism. In the context of human genomics, WGS is expected to foster significant progress in the quality of healthcare [15]. In particular, genetic features and mutations are being increasingly and more effectively studied in relation to treatment of – as well as predisposition to – diseases and genetic conditions, such as Alzheimer's, diabetes and various types of cancer. Availability of fully sequenced genomes makes such testing seamless and low-cost, since complex operations can now be executed in software, in a matter of seconds [26]. This fuels the race toward cheaper, faster, and more accurate sequencing technologies. Naturally, progress in genomics and WGS prompts numerous immediate and anticipated benefits. At the same time, it amplifies security, privacy and ethical concerns that stem from unprecedented sensitivity of genomic data. The human genome contains detailed and extremely personal information, e.g., susceptibility to somatic and mental conditions as well as ethnicity and ancestry. Therefore, potential disclosure triggers fears of genetic discrimination, aka "eugenism". Moreover, due to its hereditary (cumulative) nature, the human genome encodes information beyond the individual: it also contains information about one's siblings, parents as well as other ancestors and descendants.

A genome uniquely identifies its owner, which makes anonymization and de-identification techniques ineffective [29, 32, 39, 52, 56]. Security and privacy issues in genomics have been studied by security experts [5, 39] as well as biologists, ethicists [28, 32, 48] and policy-makers [47]. We refer to [3] for a comprehensive overview.

Risks of disclosure motivate the need for secure storage and testing of human genomes. In particular, there is a need for genetic testing by physicians and/or laboratories without full access to individuals' genomes. In an idealized future setting, genetic tests should only disclose the required minimum amount of information. At the same time, genomics and biotechnology companies often treat test details as trade secrets, since they represent valuable intellectual property [24, 25]. For instance, Myriad Genetics recently attempted to patent two human genes, which, if mutated, are associated with significantly increased risk for breast and ovarian cancers [45]. The legality of these patents was recently challenged and the US Supreme Court has declared them invalid [54]. This decision was based on the premise that, owing to its natural occurrence, no part of a genome is patentable. We believe that, because of this landmark decision, the commercial sector will have no choice but to keep its test specifics (e.g., genetic markers) proprietary. Thus, while the outcome of a genetic test might be made available to the patient and/or the testing laboratory, test specifics would be kept confidential by the latter.

These challenges extend to the emerging field of *personalized medicine*, i.e., the practice of tailoring diagnoses, presymptomatic examinations, and treatments to the precise genetic makeup of individual patients. Already today, a

WPES'13, November 4, 2013, Berlin, Germany.

ACM 978-1-4503-2487-4/13/11 ...$15.00.
http://dx.doi.org/10.1145/2517840.2517849 .

number of companies (e.g., 23andme, i-gene and Knome) provide customers with detailed reports about their predisposition to diseases and conditions. Several drugs (e.g., for treating cancer, HIV and leukemia) are coming to market accompanied by related genetic tests, that are necessary to assess correct dosage and/or expected effectiveness [1, 10, 46]. Again, while details of such tests might be proprietary, patients are strongly dis-incentivized from submitting their entire genomes for testing.

Motivated by the above discussion, this paper explores efficient cryptographic protocols for genomic testing that satisfy aforementioned privacy requirements. Specifically, we focus on a setting where one party, Bob, holds a copy of his digitized genome and the other, Alice, holds a (possibly non-contiguous) substring that might occur in a certain location of Bob's genome. The specific problem at hand is: *how to allow Bob to test for the presence of Alice's substring at a certain location in his genome, such that Bob learns nothing about the substring (including its position and its size), while Alice learns nothing about the genome?*

We address this problem by building a novel cryptographic primitive, called **Size- and Position-Hiding Private Substring Matching (SPH-PSM)** Despite its genomics-based motivation, SPH-PSM is appealing in any substring-testing setting where size and position (and, clearly, the contents) of the partial substring should be concealed, since both values can leak information about the *nature* of the test.

We also describe a prototype of SPH-PSM and demonstrate its practicality via a thorough performance evaluation, which shows that many genomics tests can be conducted today in under a minute.

Paper Organization: Next section presents the problem statement. Then, Section 3 overviews related work. Section 4 describes and analyzes the SPH-PSM protocol. Next, Section 5 presents efficient instantiations of SPH-PSM and introduces several optimizations. Section 6 discusses further extensions and variations. Finally, Section 7 concludes the paper and Appendix A presents a brief genomics primer.

2. PROBLEM STATEMENT

The quest for Predictive, Preventive, Participatory, and Personalized (P4) medicine [33] has been one of the main motivating factors in genomics research. With the advent of low-cost sequencing technologies, the natural next step is to provide physicians and testing facilities with computational means to query, correlate, and analyze entire digitized genomes [55].

One prominent challenge is where and how to store a digitized genome, i.e., 3.2 billion letters. This issue is rife with privacy and trust considerations. As noted in [5], individuals should ideally retain ownership of their sequenced genome and selectively allow partially trusted third parties (such as physicians, clinicians and testing facilities) to query it.

Privacy and ethical concerns prompt the need for secure genomic tests that offer simultaneous confidentiality of the individual's genome and test specifics. Since data obfuscation and anonymization tools are ineffective in the genomic context [29, 32, 39, 52], secure computation techniques are needed to realize privacy-preserving versions of these tests, whereby only the test result is disclosed to one or both participants. This presents some challenges: First, secure computation techniques must contend with the sheer size of the genome, which makes it crucial to maximize precomputation and minimize protocol input size. Furthermore, each current genetic test needs to be mapped to a function with output conveying nothing beyond the test outcome. For example, a testing facility may want to check for the presence of a few DNA markers in a patient's genome to determine correct dosage for a blood thinning drug: a privacy-preserving version of this test would disclose nothing beyond whether this patient has these exact markers.

Specifically, the test should not leak:

1. ***Position(s)*** of tested markers,
2. ***How many*** markers are being tested, and
3. The ***subset*** of matched markers, in case of an overall negative result.

In the genomic setting, where the number of current genomic tests is relatively small, the size- and position-hiding are particularly important. This is because disclosure of the number or positions of markers could allow the adversary to infer test specifics. As discussed in Section 3, satisfying aforementioned three requirements is an open problem, which we address in this paper.

3. RELATED WORK

Related work falls into several categories described separately below.

Secure Genomics. Motivated by extreme sensitivity of DNA data, the security research community proposed some cryptographic techniques for secure computation on short DNA *fragments*, such as: searching [7, 9, 50], computing distance between snippets [35, 53] and related functionalities [21, 37]. With the advent of affordable technologies for WGS, focus shifted to protocols that can scale to the entire genome. In particular, [5] introduced protocols for paternity testing and personalized medicine. Similar to our work, it is assumed that an individual retains control of (i.e., securely stores) his digitized genome. The protocol for secure personalized medicine testing in [5] involves (i) a patient, on input her genome, and (ii) a testing facility, on input a list of DNA mutations, along with their corresponding positions. The testing facility needs to check for the presence of these markers in the patient's genome. At the end of the interaction, the patient has no output, while the testing facility learns which markers appear in the patient's genome. The construction in [5] is based on the Private Set Intersection (PSI) concept [22] and thus has the following limitations:

- If the patient's genome contains only a subset of tested mutations, this subset is revealed to the testing facility, hence violating the goal of only disclosing the test outcome.

- The number of tested markers is revealed to the patient. This information might be enough to (partially or entirely) disclose the nature of the test, thus potentially violating the requirement of hiding test specifics from the patient.

Subsequently, [17] extended the work in [5] by implementing privacy-preserving ancestry (e.g., paternity) testing on Android devices and demonstrating its current viability. However, these results exhibit the same limitations as [5]. The

$a \leftarrow_{\$} \mathcal{A}$	Variable a chosen uniformly at random from $\mathcal{A}$
Σ=$\{A, G, C, T, -\}$	DNA Alphabet ('$-$' denotes deletion)
$p = \{\Sigma\}^m$	Potential substring held by Alice, of length m
$t = \{\Sigma\}^n$	String held by Bob, of length n
$i \in [1, m]$, $j \in [1, n]$	Indices of characters in p and t, respectively
p_i, t_j	Elements in positions i and j, of p and t, respectively
$t[j : x]$	Substring in t starting at t_j, of length x
s	Presumed starting location of p in t
λ	Security parameter
AddHomEnc	An IND-CPA additively homomorphic cryptosystem
PK, SK	Bob's private and secret key for AddHomEnc
$E(\cdot), D(\cdot)$	Encryption (with PK) and decryption (with SK) in AddHomEnc
$\mathbb{G}$	Plaintext group of E
$q = ord(\mathbb{G})$	The order of the plaintext group
$E(a) \cdot E(b) = E(a + b)$	Multiplication of two ciphertexts resulting in addition of two plaintexts
$E(a)^c = E(a \cdot c)$	Exponentiation of a ciphertext resulting in multiplication of plaintext by constant
$\perp$	No output
$\equiv^c$	Computational indistinguishability

Table 1: Notation used throughout the paper.

work in [11] proposed to secure biomedical data using cryptographic hardware, and [36] used homomorphic encryption to perform scientific investigations on integrated genomic data. Finally, [13] proposed techniques to securely map and align human genomic sequences to a reference genome, while outsourcing computation to a hybrid cloud.

Secure Pattern Matching. There are a few cryptographic techniques for Secure Pattern Matching (SPM) [6, 23, 30, 31], where one party (P_1) holds a pattern and the other party (P_2) holds a text string. P_1 learns where the pattern appears in the text, without revealing it to P_2, or learning anything else about P_2's input. However, the size of P_1's pattern is always revealed to P_2. Although [31] sketched out a way to hide the pattern size by means of wildcard padding, the upper bound on the size is still revealed. Plus, supporting wildcards causes a communication and computational performance increase from linear to multiplicative in the size of the text (n) and the size of the pattern (m). In the genomic setting, even a pattern of length 4 would result in around 12 billion modular exponentiations. In order to completely hide the pattern size, communication and computational complexity further increases to $O(n^2)$.

Moreover, SPM actually reveals *all* occurrences of P_1's pattern in P_2's string. Therefore, it is not well-suited for the problem at hand, since, our setting only needs a binary output indicating whether a substring, representing the marker(s) to be tested, appears in a larger string (i.e., the patient's genome) at some specific position(s). It is possible to extend SPM to only disclose the presence of a contiguous substring in a string at a specific location, e.g., by modifying parties' inputs from a sequence of letters to a sequence of hash(letter||position). However, there is no straightforward way to adapt SPM to match *non-contiguous* substrings, except for surrounding the pattern with single-character wild cards, which (as noted earlier) considerably increases protocol complexity.

Based on the above discussion and to the best of our knowledge, no SPM technique can efficiently handle genomic-scale inputs. Moreover, there are scarcely any SPM implementations; one is described [6]. Whereas, we report on the performance of an actual prototype which confirms the practicality of the proposed SPH-PSM technique.

Input-Size Hiding. There are only a few constructions that support hiding input size in secure computation protocols. Ishai and Paskin [34] did so in the generic context of branching programs: one party can evaluate a program on some encrypted input, in such a way that the size of the program is not revealed to the other party. In [2], Ateniese et al. showed that the assumption that secure multi-party computation necessitates revealing input sizes does not always hold. [2] demonstrated a Size-Hiding Private Set Intersection (SHI-PSI) protocol where the size of the set held by the party receiving the intersection (client) is not disclosed. Although somewhat relevant to the problem at hand, SHI-PSI cannot be used for private substring matching, since, as discussed above, any known linear reduction to PSI leaks the subset of matching mutations. Note, however, SHI-PSI could be reduced to substring matching (only for contiguous substrings) with *quadratic* computation and communication complexities: the text holder could encode each possible substring as a set element, thus creating a set with n^2 elements; meanwhile, the pattern could be represented as a single element set, and substring matching can be executed as a set intersection. Finally, Lindell et al. [38] recently presented some feasibility results on hiding input size in secure computation, based on Fully Homomorphic Encryption (FHE).

4. SPH-PSM

We now introduce a technique for private genomic testing, whereby a testing facility checks for the presence of a substring or a pattern in a patient's genome, such that the latter learns the outcome and no other knowledge is obtained by either party. We do so by introducing a cryptographic primitive called Size- and Position-Hiding private Substring Matching (SPH-PSM). After defining privacy requirements, we present a generic construction based on additively homomorphic encryption and analyze its security.

4.1 Definitions & Notation

Our **notation** is reflected in Table 1.

Definition 1 (SPH-PSM). Size- and Position-Hiding Private Substring Matching (SPH-PSM) involves two parties: Alice, on input $((p = p_1, ..., p_m), s)$, and Bob, on input $(t = t_1, ..., t_n)$:

$$F_{\text{SPH-PSM}}((p,s),t) \rightarrow (\perp, b), \text{where } b = \begin{cases} 1 \text{ iff } t[s:m] = p \\ 0 \text{ otherwise} \end{cases}$$

Correctness. If both parties are honest and run on correct input (as above), at the end of the protocol, Bob outputs 1 iff $t[s:m] = p$ and 0 otherwise (except with negligible probability).

Notation used throughout the rest of the paper is reflected in Table 1.

Adversarial Model. We use standard security models for secure two-party computation and assume the Honest-but-Curious (HbC) model. The term *adversary* refers to insiders, i.e., protocol participants. Outside adversaries are not considered, since their actions can be mitigated via standard network security techniques. Protocols secure in the HbC adversarial model assume that all parties faithfully follow all protocol specifications. However, during or after protocol execution, any party might (passively) attempt to infer additional information about the other party's input.

We argue that, in the genomic testing setting, HbC security is sufficient. Genomic testing usually takes place in a medical lab or a physician's office and we therefore expect some degree of trust between the patient and the testing lab. Thus, we envision no incentive for participants to arbitrarily deviate from the protocol. Also, due to their sensitivity, genomic tests can be audited and there might be legal consequences for malicious behavior by either party. Furthermore, in the event of a testing facility breach and/or data loss, privacy of the genome holder remains guaranteed.

The HbC model is formalized by considering an ideal implementation where a trusted third party (TTP) receives the inputs of both parties and outputs the result of the desired function. Security in this model requires that, in the real implementation of the protocol (without a TTP), each party does not learn more information than in the ideal implementation. We introduce simulation-based privacy definitions below.

NOTE: Recall that, in our setting, Alice plays the role of the testing laboratory and Bob is the individual in possession of a digitized genome.

Alice's Privacy. Let $View_B((p,s),t)$ be a random variable representing Bob's view during the real execution of SPH-PSM. We say that SPH-PSM guarantees Alice's privacy if there exists a Probabilistic Polynomial Time (PPT) algorithm B^* such that, given Bob's output b, the following holds:

$$\{B^*(t,b)\}_{((p,s),t)} \equiv^c \{View_B((p,s),t)\}_{((p,s),t)}$$

That is, for all possible inputs, with the knowledge of the output bit b and Bob's input, B^* can efficiently simulate the view of Bob.

Bob's Privacy. Similarly, let $View_A((p,s),t)$ be a random variable representing Alice's view during the real execution of SPH-PSM. We say that SPH-PSM guarantees Bob's privacy if there exists a PPT algorithm A^* such that, given $n = |t|$:

$$\{A^*((p,s),n)\}_{((p,s),t)} \equiv^c \{View_A((p,s),t)\}_{((p,s),t)}$$

Although we require knowledge of text length for efficient simulation, in many applications this variable is public or fixed, e.g., 3.2 billion for human genomes.

Additively Homomorphic Encryption. We assume the existence of an efficient additively homomorphic public-key cryptosystem with indistinguishability under CPA attacks (IND-CPA). To ease presentation, we denote it as AddHomEnc. Recall that there are several AddHomEnc instantiations, including: Paillier [44], Dåmgard-Jurik [16], as well as the Additively Homomorphic ElGamal variant [19].

For each instantiation application of homomorphic operations vary slightly. As described in Table 1, we denote the homomorphic addition operation as multiplication of ciphertexts. Further, we denote multiplication by a constant by the exponentiation of a ciphertext.

4.2 Proposed Construction

We now present an SPH-PSM protocol that involves Bob on input a string t (i.e., a genome), and Alice who holds a substring p (of length m) that might occur at location s in t. We later extend this protocol to support multiple starting locations. At the end of the interaction, Bob learns nothing about Alice's substring, *not even its length or intended position*, while Alice learns the test outcome, i.e., $b = 1$ iff $t[s:m] = p$ and $b = 0$ otherwise.

Protocol. Figure 1 illustrates the protocol. It relies on an additively homomorphic cryptosystem AddHomEnc, composed by $KeyGen(1^\lambda)$, $E(\cdot)$, and $D(\cdot)$. It assumes that Bob has previously generated a keypair (PK, SK) for AddHomEnc, as part of *KeyGen*.

First, Bob encrypts, under its own public key, each nucleotide of the genome and sends Alice the resulting ciphertexts. Starting at position s, Alice homomorphically adds each encrypted nucleotide to the encryption (under PK) of the inverse of each substring character. If the corresponding plaintexts match, the product of the two ciphertexts is an encryption of zero, due to the homomorphic property of AddHomEnc. To guarantee privacy, each product is re-randomized by exponentiating it to a random value. Then, Alice multiplies all products so that, if the substring is found in Bob's genome, the result would still be an encryption of zero. Otherwise, it corresponds to an encryption of a random number. Finally, this cumulative ciphertext (denoted as *acc*) is sent to Bob. Upon decryption, Bob learns the test outcome: positive if the ciphertext decrypts to zero and negative otherwise.

Complexity. *Communication* overhead amounts to $O(n)$ ciphertexts, i.e., $\approx$ 3.2 billion encrypted nucleotides. Although this might seem overwhelming, we discuss how to reduce online costs in Section 6.4. On the other hand, *computational* complexity is minimal, since the only operation that involves all n nucleotides is encryption of Bob's genome. This needs to be done only once, ahead of time, for all possible tests. In fact, it could even be done by the sequencing lab, at sequencing time. Whereas, *online* computational complexity is linear in the size of the substring, i.e., $m = |p|$. Specifically, Alice needs to perform $O(m)$ operations (where $m << n$), while Bob's overhead is constant — one decryption.

Finally, Alice only performs computation on ciphertexts et_j for $s \leq j < s+m$. All other ciphertexts can be discarded.

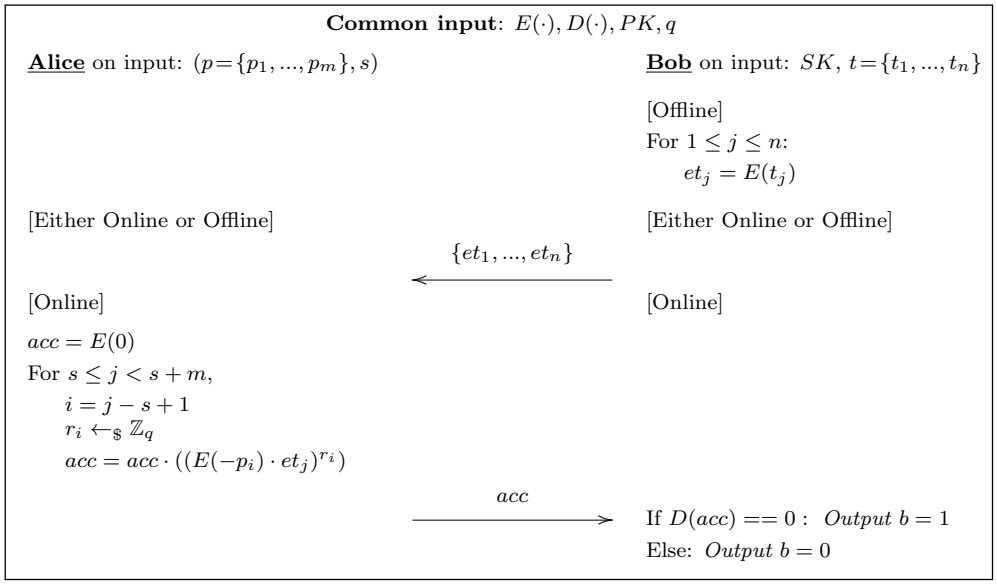

Figure 1: Base-line SPH-PSM Protocol.

Also, Alice's computation can start as soon as et_s is received. Therefore, Alice requires only a negligible amount of local *storage*.

4.3 Security Analysis

Theorem. If AddHomEnc is an IND-CPA additively homomorphic cryptosystem, the protocol in Figure 1 correctly computes SPH-PSM with both Bob's and Alice's privacy.

Proofs. We now show that the protocol satisfies correctness and both parties' privacy, as defined in Section 4.3.

Correctness. We show that $b = 1$ if and only if $t[s:m] = p$ except for probability negligible in λ. First, we note that:

$$t[s:m] = p \iff \forall_{i=1,...,m}\ t_{s+i-1} = p_i \Longrightarrow \sum_{i=1}^{m}(t_{s+i-1} - p_i)^{r_i} = 0$$

Furthermore, since:

$$acc = \prod_{i=1}^{m}(E(t_{s+i-1}) \cdot E(-p_i))^{r_i} = E\left(\sum_{i=1}^{m}(t_{s+i-1} - p_i)^{r_i}\right)$$

it follows that:

$$t[s:m] = p \Longrightarrow \textstyle\sum_{i=1}^{m}(t_{s+i-1} - p_i)^{r_i} = 0 \iff acc = E(0) \iff b = 1$$

To complete the proof, we only need to show that:

$$\sum_{i=0}^{m}(t_{s+i-1} - p_i)^{r_i} = 0 \Longrightarrow t[s:m] = p$$

Note that, if p_i does not match t_{s+i}, then the i-th element in the sum is a random group element r'_i.

$$t_{s+i-1} \neq p_i \Longrightarrow (t_{s+i-1} - p_i)^{r_i} = r'_i$$

Thus, the sum comprises m random group elements. However, it equals zero with probability $1/q$, which is negligible in λ. Hence, if the sum is zero, then it must hold that $t[s:m] = p$, with overwhelming probability.

Alice's Privacy. To show that Bob's view can be efficiently simulated, we construct a simulator B^*. On input of (t, b), B^* outputs the real transcript, except for acc – the only value received by Bob– which B^* computes as: $acc = E(0)$ if $b = 1$ and $acc = E(r)$ for $r \leftarrow_\$ \mathbb{Z}_q$, otherwise. Note that acc is distributed identically to the real execution, since, as noted above, if $b = 0$, then acc is the encrypted sum of one or more random group elements and/or many zeros, and is itself an encryption of a random group element.

Bob's Privacy. To show Bob's privacy, we construct a simulator A^*. On inputs $((p, s), n)$, A^* first outputs $\{et'_1, ..., et'_n\}$ where $et'_j = \{E(r_j)\}$ and $r_j \leftarrow_\$ \mathbb{Z}_q$. Then, A^* outputs the rest of the view, as usual. Based on indistinguishability of AddHomEnc, it holds that: $\forall j\ et'_j \equiv^c et_j$, hence the view and the simulation are computationally indistinguishable.

4.4 Timing Attacks

Alice's and Bob's privacy properties discussed above guarantee that each party only learns what it is intended to learn, based on the information (i.e., protocol messages) exchanged. However, they do not take into account auxiliary means of information leakage, such as the timing channel.

It is easy to see that Alice's computation in the online phase of the protocol is proportional to m – search substring size. Thus, assuming negligible message transmission delay and knowledge of the Alice's computing platform, Bob (or anyone observing the protocol) could estimate m.[1]

There are several ways to counter such timing attacks. One trivial counter-measure is for Alice to pipeline (or parallelize) transfer of the encrypted genome, i.e., $\{et_1, ..., et_n\}$, with the computation of acc in the loop of Figure 1. Since $m << n$, $O(m)$ computation overhead at Alice is dominated by the $O(n)$ communication overhead incurred by transfer of the encrypted genome. Though this is easy to achieve, Alice would have to wait to reply with acc until all cipher-

[1]Note that timing attacks against Bob are not meaningful in our context.

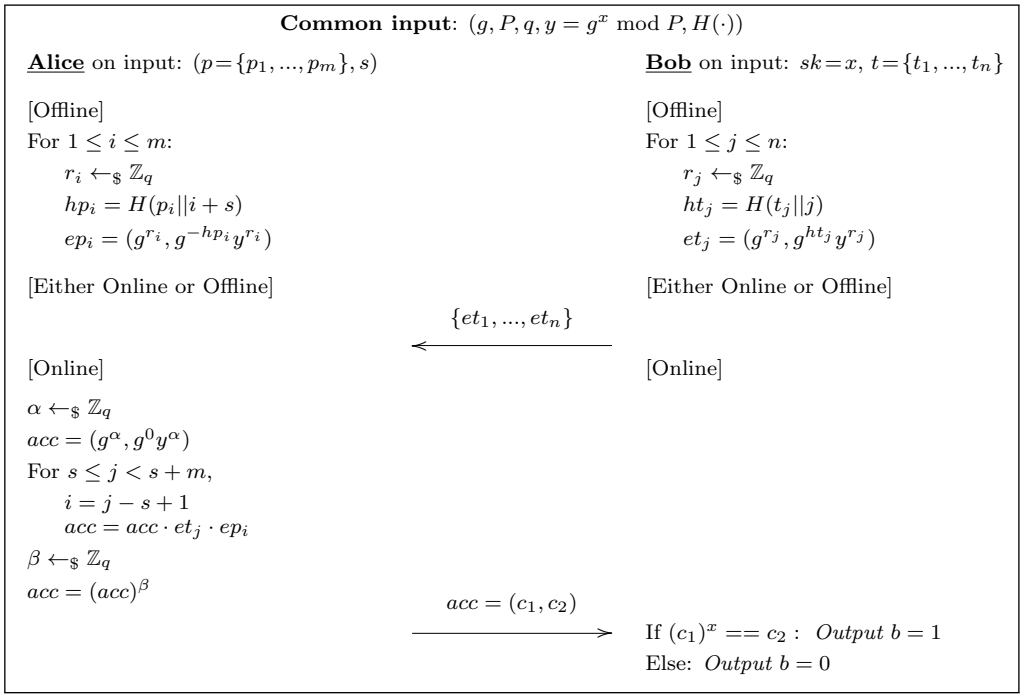

Figure 2: AH-ElGamal based SPH-PSM. (Computation is done mod P).

texts are received, regardless of the starting location of the search substring.

Alternatively, we could consider transfer of the encrypted genome as part of the protocol's offline phase. Then, the online phase begins with Alice performing $O(m)$ operations and no other party can infer m since it is not privy to Alice's *actual* starting time. This might, in fact, reflect a real-world setting where a testing facility receives encrypted genomes from many patients and batches computation.

Finally, Alice could delay sending the final protocol message containing *acc* for a certain period. If this delay matches the time to compute a pattern of length m', Alice can effectively pad m to m'. With the optimized version of our protocol (see Section 5.2), Alice's $O(m)$ operations reflect inexpensive modular multiplications, which can be completed, for $m \leq 10^8$, in under one minute. Thus, m' can be several orders of magnitude greater than m, without unduly affecting the overall run-time of the protocol.

5. SPH-PSM IN PRACTICE

We now discuss some practical considerations for implementating SPH-PSM.

5.1 Instantiating AddHomEnc

As discussed in Section 4, SPH-PSM relies on availability of AddHomEnc, i.e., IND-CPA-secure additively homomorphic encryption. There are several suitable candidates, including Paillier [44], Dåmgard-Jurik [16], and Okamato-Uchiyama [42].

However, it is easy to see that SPH-PSM does not actually require ability to decrypt arbitrary ciphertexts. In fact, for the purposes of SPH-PSM, it suffices to test whether a ciphertext is an encryption of zero. This allows us to consider the Additively Homomorphic ElGamal variant (AH-ElGamal) as one of the choices.

AH-ElGamal involves the following algorithms:

- *Key Generation:* On input of security parameter λ, select public parameters (g, P, q) for primes P, q and g generator of subgroup of $\mathbb{Z}_P$ of order q. (Note: we use P instead of more customary p, since, in our notation, p denotes Alice's substring). Private key SK is $x \leftarrow_\$ \mathbb{Z}_q$ and public key PK is $y = g^x \bmod P$.
- *Encryption:* $E_{PK}(m) = (c_1, c_2)$, where $c_1 = g^r \bmod P$ and $c_2 = g^m y^r \bmod P$, for $r \leftarrow_\$ \mathbb{Z}_q$.
- *Testing for Encryption of Zero:* $D_{SK}(c_1, c_2) = 0$ if and only if $c_2 = (c_1)^x \bmod P$.

There is also an Elliptic Curve-based ElGamal variant (EC-ElGamal) that is additively homomorphic [51]. It involves the following algorithms:

- *Key Generation:* On input of security parameter λ, select public parameters (P, e, G, η), where e is an elliptic curve over finite field $\mathbb{GF}(P)$, η is the order of curve e, and G is the generator point of e. SK is integer $x \in \mathbb{GF}(P)$, and PK is $Y = xG$.
- *Encryption:* $E_{PK}(m) = (R, S)$, where $R = rG$ and $S = M + rY$, for $r \in [1, \eta - 1]$.
- *Decryption:* $D_{SK}(R, S) = -xR + S$.

Naturally, to efficiently realize SPH-PSM, we need an AddHomEnc instantiation that meets the following criteria: (1) ability to test for encryption of zero, (2) small ciphertext size, (3) fast encryption, and (4) fast homomorphic addition operations.

Actually, the "weight" of such factors depends on whether transfer of Bob's encrypted genome is part of the offline phase. If so, complexity of SPH-PSM is clearly dominated by Alice's $O(m)$ computation overhead. Thus, the preferred instantiation is AH-ElGamal due to its computational efficiency. Since random exponents can be taken from a sub-

group (e.g., 160-bit), encryptions and re-randomizations can be computed efficiently, e.g., relative to Paillier.

Conversely, if transferring the encrypted genome is part of the online phase, transfer dominates complexity of SPH-PSM. Thus, the preferred instantiation is EC-ElGamal, since group elements are much smaller (160-bit), as opposed to 1024-bit in AH-ElGamal and 2048-bit in Paillier, using the same security parameters.

5.2 ElGamal-based SPH-PSM

We now present an ElGamal-based instantiation of SPH-PSM and introduce some optimizations. We focus on reducing computational complexity of the online phase. Although we describe this instantiation in terms of AH-ElGamal, the discussion applies to EC-ElGamal as well.

In AH-ElGamal, multiplying two ciphertexts (adding two corresponding plaintexts) is markedly more efficient than exponentiating a ciphertext with a constant (multiplying the corresponding plaintext by the constant). Therefore, we modify the basic protocol of Figure 1 such that Alice's online phase only involves multiplications. Rather than encrypting a raw nucleotide, Bob encrypts the hash of the nucleotide along with its corresponding position in the genome. To this end, we need a hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$ (modeled as a random oracle)[2].

The resulting protocol is shown in Figure 2. Alice performs $O(m)$ modular multiplications and only $O(1)$ modular exponentiations during the online phase. This greatly reduced online overhead also helps thwart timing attacks discussed in Section 4.4.

5.3 Performance Evaluation

Protocols proposed in this paper have been implemented in `C++`, and tested on a desktop machine running Ubuntu 12.10, with an Intel i7-3770 3.4GHz quad-core CPU and 16GB of RAM. We implemented 1024-bit AH-ElGamal using the gmp [27] library and 160-bit EC-ElGamal – using the OpenSSL library [43]. We break up the measurements as follows:

1. Genome Encryption – time for Bob to encrypt his genome.
2. Data Transfer – time for Alice to download Bob's genome.
3. Substring Pre-processing – time for Alice to pre-process her substring.
4. Online Phase – time elapsed after genome is received, until Bob outputs the result, i.e., b. (This phase is dominated by Alice's computation).

Experiments were conducted with Alice's substring size ranging from 10 to $3 \cdot 10^9$ letters. For tests taking longer than several hours, run-times have been estimated by performing tests on a "smaller" genome (10^7 nucleotides). This is justifiable since tested protocols incur linear complexities.

We also note that our experiments were conducted rather conservatively, using a single thread on a regular (mid-range) desktop machine. More realistic scenarios would involve machines with multiple cores, much bigger RAM and higher-end CPU-s.

First, we measured genome encryption: it took approximately 115 hours using AH-ElGamal and an impressive $2,580$ hours using EC-ElGamal. This significant difference can be explained by the fact that AH-ElGamal is implemented using gmp, whereas, EC-ElGamal – using OpenSSL, which is markedly slower. While we do plan to optimize EC-ElGamal performance, this issue is somewhat secondary, since: (1) EC-ElGamal is used in settings where first priority is to minimize communication overhead, and (2) a genome is encrypted only once, ahead of time, and (3) encryption can be easily parallelized, using multiple cores.

Next, we estimate the time to transfer the encrypted genome. On a 1Gbps link, it takes approximately 2.7 hours with EC-ElGamal and 8.7 hours with AH-ElGamal. Due to linear complexity of SPH-PSM, transfer times over networks with different speed can be straightforwardly inferred, e.g., 27 hours with EC-ElGamal and 87 hours with AH-ElGamal over a 100Mbps link.

Finally, Table 2 reports on run-times for Alice's substring pre-processing and for the online phase dominated by Alice's computation. For the latter, we quantify the speedup due to the optimization presented in Section 5.2.

Although we report on tests conducted up to the largest possible substring, i.e., the entire genome ($3 \cdot 10^9$ letters), most genomic applications used today require testing of significantly smaller substrings. For example, prior work in [5] and [17] supported personalized medicine testing with fewer than $1,002$ markers, e.g., analysis of the *tpmt* gene common in leukemia patients, or testing for *hla-b* variants associated to sensitivity to some HIV drugs. For this class of tests, our optimized SPH-PSM protocol completes in less than minute. However, it is still feasible to securely conduct future tests with much larger substrings.

6. EXTENSIONS

We now discuss some natural extensions to the basic SPH-PSM protocol.

6.1 Revealing Test Outcome to Alice

Under some circumstances, it might be necessary for both Bob and Alice to learn the test outcome. For example, "Alice" might actually be a medical laboratory and it could be in the patient's best interest to reveal the test outcome to a specialist as soon as possible.

In the HbC model,we can trivially extend the basic SPH-PSM protocol to allow Bob to communicate b to Alice. Nonetheless, we leave it for future work to explore the use of threshold cryptosystems to provably enforce mutual output in SPH-PSM, e.g., by relying on threshold ElGamal [18] or threshold Paillier [16, 20].

6.2 Fixed-Size Wildcards and Non-Contiguous Substrings

Alice's substring might contain fixed-length wildcards or it might be non-contiguous. There is no real difference between a substring such as "ACG??TAAC" (where "?" is a single-character wildcard) and a two-part (non-contiguous) substring "ACG" and "TAAC", with each part starting at some locations s_1 and s_2, that are arbitrary number of positions apart, e.g., $s_2 = s_1 + 5$.

It is easy to see that our basic protocol is independent of contiguity of Alice's substring. Computational complexity remains $O(m)$.

6.3 Multiple Substrings/Starting Locations

[2]Without a hash function, there can be cross cancellation between positions.

	AH-ElGamal			EC-ElGamal		
$\mathbf{m = \|p\|}$	**Online**	**Online Optimized**	**Substring Prepr.**	**Online**	**Online Optimized**	**Substring Prepr.**
10	0.76ms	0.1ms	0.46ms	12.71ms	0.78ms	7.05ms
10^2	4.51ms	0.22ms	4.05ms	66.15ms	3.77ms	36.9ms
10^3	28.72ms	0.68ms	20.35ms	635.99ms	32.64ms	317.62ms
10^4	291.12ms	6.02ms	168.28ms	6.41s	318.65ms	3.17s
10^5	2.86s	60.18ms	1.63s	1m 7s	3.28s	31.95s
10^6	28.61s	647.05ms	16.9s	10m 35s	31.85s	5m 17s
10^7	4m 45s	6.25s	2m 41s	1h 45m	5m 19s	52m 55s
10^8	47m 37s	1m 2s	26m 53s	17h 38m.	53m 17s	8h 49m
10^9	7h 56m	10m 25s	4h 28m	7d 8h	8h 52m	3d 16h
$3 \cdot 10^9$	23h 48m	31m 16s	13h 26m	22d 1h	1d 2h	11d 54s

Table 2: SPH-PSM: computation time for varying substring lengths.

In some applications, Alice might not know the exact starting position of a particular search substring: it could begin in several (v) possible positions. This situation is typical in genomic testing since alignment of the patient's genome may not always be exactly precise. We can easily extend SPH-PSM such that Alice only learns whether ***any*** one of these positions match. The resulting extension is a trivial variation of the basic version. The resulting protocol only incurs $O(m * v)$ computation overhead, at Alice's side, as compared to $O(m)$ in the basic protocol; communication and Bob's computation overheads are unaltered.

A similar case occurs when, even if the exact starting position is known, a particular position in a search substring can be ***one of several choices***. For example, "GA[G,T]TACA" denotes that position 3 in the substring can be either G or T. This is clearly distinct from a single-character wildcard, since neither A nor C is allowed in this position. The natural way to deal with such disjunctive conditions is to try to match two substrings: "GATTACA" and "GAGTACA" starting at the same position. The very same protocol extension applies here.

Yet another variation occurs whenever multiple distinct substrings $(p_1, \ldots, p_w)$ are matched in parallel or disjunctively. In this case Bob learns ***how many*** matches occur. This case is slightly different from that of non-contiguous substrings or wildcards mentioned above, where Bob only learns the result of a single test. Again, the same extension handles this case. Communication and Bob's computation complexities are unaltered, while Alice's computation overhead would amount to $O(\sum_{i=1}^{w} |p_i|)$, as compared to $O(|p| = m)$ in the basic version.

6.4 Reducing Data Transfer Time

The basic protocol entails Bob sending the entire letter-by-letter encrypted genome to Alice. This constitutes the most costly part of the protocol – about $3.2 \cdot 10^9$ ciphertexts. With EC-ElGamal, where each ciphertexts is a 320-bit value, transferring the encrypted genome translates into roughly 100GB of data.

However, this massive data transfer needs to happen only once for many tests that Bob (e.g., a given patient) conducts with Alice (e.g., a particular medical facility). Also, while transmitting encrypted genomes in real time might be unfathomable today, it will likely become realistic in not-so-distant future. Plus, most genomic tests are not typically spontaneous, i.e., patients plan and schedule them in advance. Consequently, transfer of the encrypted genome can take place incrementally over some period preceding the actual test. In fact, a testing facility could interact with many patients at a time, gradually uploading their encrypted genomes. Once a given patient's genome is fully uploaded, the actual protocol begins.

Physical Transfer. For some tests, Bob might physically visit the testing facility (Alice). In that setting, Bob might be asked to bring along a dedicated device, e.g., a USB stick or a portable flash drive, containing the entire encrypted genome. Such passive devices capable of storing over 1TB of data are readily available today and cost well below US$100. Once plugged into Alice's computing equipment, much faster transfer rates apply: for example, USB 3.0 offers transfer rates of up to 4.8Gbps, which translates into about 3 minutes for transferring the encrypted genome.

Alternatively, Alice could simply read the desired m ciphertexts directly from Bob's drive and perform computation on them, without bothering with any other data. This would clearly minimize communication delay. However, care must be taken to prevent attacks that might exploit the portable drive's logs or other data to later infer locations that have been read by Alice.

Leveraging Cloud Storage. Another possibility is to use the cloud. By storing Bob's encrypted genome at a cloud provider, transfer of the encrypted genome would need to be performed only once, for all possible tests and testing facilities. The cloud provider can offer both higher bandwidth and availability. However, if the cloud provider is also trusted not to collude with Bob, Alice could avoid bulk transfer and selectively fetch only desired ciphertexts. (A collusion would reveal the size and position of the substring held by Alice.)

One potential concern, even without any collusion, is that the cloud provider would learn the size and position of Alice's substring. Fortunately, the latter can be mitigated by letting Alice and Bob pre-agree on a common secret that Bob could later use to shuffle encrypted nucleotides before uploading to the cloud. This way, ordering of ciphertexts (but not m – the substring size) would be obscured from the cloud provider.

Alternatively, Alice could rely on cryptographic techniques for Private Information Retrieval (PIR) [14] to securely retrieve m desired ciphertexts from the cloud. Despite PIR's computational overhead, recent results [8, 40] show that PIR can be efficiently adapted to cloud settings where a static database is maintained in a MapReduce cluster. However,

the size of Alice's substring would be revealed to the cloud provider.

6.5 Coping with (Some) Malicious Input

We now consider the case where malicious Bob tries to guess the specifics of Alice's test, by verifying its guess of Alice's input. Let (p, s) denote Alice's input and (p', s') – Bob's guessed version. While faithfully following all SPH-PSM protocol steps, Bob inputs a concocted genome t' where $t'[s' : m'] = p'$ and $m' = |p'|$. In other words, the substring in t' starting at position p' is set to s'. (We assume, without loss of generality, that s and s' are contiguous substrings). For all other positions, Bob sets $t_j = X$ for some $X \notin \{A, C, G, T, -\}$. Then, Bob uses t' as input for SPH-PSM with Alice. It is easy to see that, if *acc* decrypts to zero, Bob learns that $(p, s) = (p', s')$.

On one hand, in the HbC model, Bob is assumed not to deviate from the protocol, which should rule out this guessing attack. On the other hand, while HbC is clear with respect to participants scrupulously following all protocol steps, it is somewhat murky with regard to the validity or goodness of participants' input. To err on the side of caution, we can address Bob's input validity (and hence guessing attacks such as the one above) by indirectly integrating the genome sequencing lab into the domain of SPH-PSM.

Specifically, at sequencing time, the genome sequencing facility (denoted by Charlie) could offer some additional services beyond supplying a digitized genome:

- Encrypt the genome, one nucleotide at a time, under Bob's public key to produce $et_1, ..., et_n$. This is a reasonable service for Charlie to provide, since it anyhow learns Bob's genome and must communicate it back securely.
- Sign the encrypted genome $(et_1, ..., et_n)$ as a whole. Similarly, to prevent any kind of future disputes and provide overall integrity as well as origin authenticity, it makes sense for Charlie to sign what it produces.

Armed with these modifications we can easily amend SPH-PSM to let Bob send Alice a *signed* encrypted genome. By verifying Charlie's signature, Alice can be assured that Bob's input is a valid genome sequenced by a reputable entity – Charlie. This change has negligible effect on performance if transfer of the encrypted genome is part of the online phase. This is because Alice needs to anyway receive all ciphertexts; it can gradually hash the encrypted genome as it is being received and, at the end, verify Charlie's signature.

However, if transfer is done off-line (as in Section 6.4) or the encrypted genome is accessed piece-meal from a cloud provider, signing the entire encrypted genome is not useful. In this case – also at sequencing time – Charlie could individually sign each ciphertext (encrypted nucleotide). During the off-line phase, Alice can obtain selected ciphertext directly (from the cloud provider or from Bob's flash drive or USB stick) and easily verify that each et_j, for $s \leq j < s+m$, is accompanied by a valid Charlie's signature.

7. CONCLUSION

This paper presented a novel cryptographic primitive called Size- and Position-Hiding Private Substring Matching (SPH-PSM) that appeals to the increasingly relevant scenario of privacy-preserving genomic testing (Personalized Medicine). A prototype implementation attests to the practicality of the proposed technique.

Naturally, our work does not end here. We intend to explore the use of SPH-PSM in the context of unordered sets, i.e., to realize efficient two-party private set disjointness test, while hiding the size of one party's set. Let one party (P_1) represent its set as a polynomial $\mathcal{P}$ (with roots corresponding to set items) and sends encrypted coefficients to the other party (P_2). P_2, for each element y_j in its set, can obliviously evaluate $\mathcal{P}$ at y_j using additively homomorphic encryption, and compute $ee_j = E(r_j\mathcal{P}(y_j))$, for a random value r_j. After multiplying all ee_j values, P_2 can send the result to P_1, which, upon decryption, only learns whether the sets are disjoint.

Furthermore, we plan to extend SPH-PSM to support additional genomic tests, such as, testing for organ donor-recipients matching. We also intend to distribute an optimized open-source implementation of SPH-PSM, along with implementations of some common Personalized Medicine tests, such as: *hla-B*, *tpmt*, as well as API support for application developers.

References

[1] A. Abbott. Special section on human genetics: With your genes? Take one of these, three times a day. *Nature*, 425(6960), 2003.

[2] G. Ateniese, E. De Cristofaro, and G. Tsudik. (If) Size Matters: Size-Hiding Private Set Intersection. In *PKC*, 2011.

[3] E. Ayday, E. De Cristofaro, J.-P. Hubaux, and G. Tsudik. The Chills and Thrills of Whole Genome Sequencing. *ArXiv Report 1306.1264*, 2013.

[4] E. Ayday, M. Humbert, J. Fellay, P. J. McLaren, J. Rougemont, J. L. Raisaro, A. Telenti, and J.-P. Hubaux. Privacy-Enhancing Technologies for Medical Tests Using Genomic Data. https://documents.epfl.ch/users/a/ay/ayday/www/erman_publications/genomic_privacy.pdf, 2012.

[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 *CCS*, 2011.

[6] J. Baron, K. El Defrawy, K. Minkovich, R. Ostrovsky, and E. Tressler. 5pm: Secure pattern matching. In *SCN*, 2012.

[7] M. Blanton and M. Aliasgari. Secure outsourcing of dna searching via finite automata. In *DBSec*, 2010.

[8] E. Blass, R. D. Pietro, R. Molva, and M. Onen. PRISM: Privacy-Preserving Searches in MapReduce. In *PETS*, 2012.

[9] F. Bruekers, S. Katzenbeisser, K. Kursawe, and P. Tuyls. Privacy-Preserving Matching of DNA Profiles. http://eprint.iacr.org/2008/203, 2008.

[10] A. Burke. Foundation Medicine: Personalizing Cancer Drugs. http://is.gd/foundation_medicine, 2012.

[11] M. Canim, M. Kantarcioglu, and B. Malin. Secure Management of Biomedical Data With Cryptographic Hardware. *IEEE Transactions on Information Technology in Biomedicine*, 16(1), 2012.

[12] B. Carlson. SNPs – A shortcut to personalized medicine. *Genetic Engineering & Biotechnology News*, 2008.

[13] Y. Chen, B. Peng, X. Wang, and H. Tang. Large-Scale Privacy-Preserving Mapping of Human Genomic Sequences on Hybrid Clouds. In *NDSS*, 2012.

[14] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *FOCS*. IEEE, 1995.

[15] F. Collins and V. McKusick. Implications of the Human Genome Project for medical science. *Jama*, 285(5), 2001.

[16] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *PKC*, 2001.

[17] E. De Cristofaro, S. Faber, P. Gasti, and G. Tsudik. GenoDroid: Are Privacy-Preserving Genomic Tests Ready for Prime Time? In *WPES*, 2012.

[18] Y. Desmedt and Y. Frankel. Threshold cryptosystems. In *CRYPTO*, 1989.

[19] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on Information Theory*, 31(4), 1985.

[20] P.-A. Fouque and D. Pointcheval. Threshold cryptosystems secure against chosen-ciphertext attacks. In *ASIACRYPT*, 2001.

[21] M. Franz, B. Deiseroth, K. Hamacher, S. Jha, S. Katzenbeisser, and H. Schröder. Towards secure bioinformatics services (short paper). *Financial Cryptography and Data Security*, 2012.

[22] M. Freedman, K. Nissim, and B. Pinkas. Efficient private matching and set intersection. In *EUROCRYPT*, 2004.

[23] R. Gennaro, C. Hazay, and J. Sorensen. Text Search Protocols with Simulation Based Security. In *PKC*, 2010.

[24] Genomics Law Report. Patenting and Personal Genomics: 23andMe Receives its First Patent, and Plenty of Questions. http://preview.tinyurl.com/7ebpft9, 2012.

[25] Genomics Law Report. Some Thoughts on Myriad After the Supreme Court Argument. http://preview.tinyurl.com/bqy25wz, 2012.

[26] G. Ginsburg and H. Willard. Genomic and personalized medicine: foundations and applications. *Translational Research*, 154(6), 2009.

[27] GMP. http://gmplib.org/.

[28] D. Greenbaum, A. Sboner, X. Mu, and M. Gerstein. Genomics and privacy: Implications of the new reality of closed data for the field. *PLoS Computational Biology*, 7(12), 2011.

[29] M. Gymrek, A. L. McGuire, D. Golan, E. Halperin, and Y. Erlich. Identifying personal genomes by surname inference. *Science*, 339(6117):321–324, 2013.

[30] C. Hazay and Y. Lindell. Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. In *TCC*, 2008.

[31] C. Hazay and T. Toft. Computationally secure pattern matching in the presence of malicious adversaries. *ASIACRYPT*, 2010.

[32] N. Homer et al. Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. *PLoS Genetics*, 4(8), 2008.

[33] L. Hood and D. Galas. P4 Medicine: Personalized, Predictive, Preventive, Participatory A Change of View that Changes Everything. http://www.cra.org/ccc/docs/init/P4_Medicine.pdf, 2009.

[34] Y. Ishai and A. Paskin. Evaluating branching programs on encrypted data. In *TCC*, 2007.

[35] S. Jha, L. Kruger, and V. Shmatikov. Towards practical privacy for genomic computation. In *S&P*, 2008.

[36] M. Kantarcioglu, W. Jiang, Y. Liu, and B. Malin. A Cryptographic Approach to Securely Share and Query Genomic Sequences. *IEEE Transactions on Information Technology in Biomedicine*, 12(5):606–617, 2008.

[37] J. Katz and J. Malka. Secure text processing with applications to private dna matching. In *CCS*, 2010.

[38] Y. Lindell, K. Nissim, and C. Orlandi. Hiding the Input-Size in Secure Two-Party Computation. http://eprint.iacr.org/2012/679, 2012.

[39] B. Malin. An evaluation of the current state of genomic data privacy protection technology and a roadmap for the future. *Journal of the American Medical Informatics Association*, 12(1), 2005.

[40] T. Mayberry, E.-O. Blass, and A. H. Chan. PIRMAP: Efficient Private information Retrieval for MapReduce. In *FC*, 2013.

[41] National Center for Biotechnology Information (US). Single Nucleotide Polymorphism Database. http://www.ncbi.nlm.nih.gov/projects/SNP/.

[42] T. Okamoto and S. Uchiyama. A new public-key cryptosystem as secure as factoring. In *EUROCRYPT*, 1998.

[43] OpenSSL. http://www.openssl.org/.

[44] P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *EUROCRYPT*, 1999.

[45] A. Pollack. Justices Consider Whether Patents on Genes Are Valid. http://nyti.ms/XB7Tf9, 2013.

[46] A. Prat and J. Baselga. The role of hormonal therapy in the management of hormonal-receptor-positive breast cancer with co-expression of her2. *Nature Clinical Practice Oncology*, 5(9), 2008.

[47] Presidential Commission for the Study of Bioethical Issues. PRIVACY and PROGRESS in Whole Genome Sequencing. http://www.bioethics.gov/cms/sites/default/files/PrivacyProgress508.pdf, 2012.

[48] S. Sankararaman, G. Obozinski, M. Jordan, and E. Halperin. Genomic privacy and limits of individual detection in a pool. *Nature Genetics*, 41(9), 2009.

[49] P. Stenson et al. The human gene mutation database: 2008 update. *Genome Medicine*, 1(1), 2009.

[50] J. Troncoso-Pastoriza, S. Katzenbeisser, and M. Celik. Privacy preserving error resilient dna searching through oblivious automata. In *CCS*, 2007.

[51] O. Ugus, D. Westhoff, R. Laue, A. Shoufan, and S. A. Huss. Optimized implementation of elliptic curve based additive homomorphic encryption for wireless sensor networks. *arXiv preprint 0903.3900*, 2009.

[52] R. Wang et al. Learning your identity and disease from research papers: information leaks in Genome Wide Association Study. In *CCS*, 2009.

[53] R. Wang, X. Wang, Z. Li, H. Tang, M. Reiter, and Z. Dong. Privacy-preserving genomic computation through program specialization. In *CCS*, 2009.

[54] R. Wolf. Justices rule human genes cannot be patented. http://www.usatoday.com/story/news/nation/2013/06/13/supreme-court-gene-breast-ovarian-cancer-patent/2382053/, 2013.

[55] S. Young. Knome Software Makes Sense of the Genome. http://www.technologyreview.com/news/428179/knome-software-makes-sense-of-the-genome/, 2012.

[56] X. Zhou, B. Peng, Y. Li, Y. Chen, H. Tang, and X. Wang. To Release Or Not To Release: Evaluating Information Leaks in Aggregate Human-Genome Data. In *ESORICS*, 2011.

APPENDIX

A. GENOMICS PRIMER

This section provides some background on genomics.

Genomes carry hereditary information needed to build and maintain an organism. Aside from certain kinds of viruses, genomes are encoded in double-stranded DeoxyriboNucleic Acid (DNA) molecules, i.e., two long polymer chains of four units called nucleotides. A nucleotides is represented by one of the four letters: A, C, G, and T. A human genome consists of $\approx$ 3.2 billion nucleotides.

Whole Genome Sequencing (WGS) is the process of determining the complete and exact DNA sequence of an organism's genome. Today, sequencing techniques extract, from a DNA sample (e.g., saliva, hair, nails blood and skin flakes), short DNA reads with hundreds of nucleotides, that are then analyzed and aligned to a so-called *reference genome*. This allows progressive reconstruction of the whole genome. Data produced by sequencing machines is usually in the form of a set of aligned strings, with associated accuracy scores. Thus, in order to represent a genome as input to SPH-PSM, we need to convert it to a single-string representation where each character in the string corresponds to the letter in the sequenced genome at the same offset.

Indels are occasional, naturally occurring insertions or deletions in genomes. A deletion happens when one or more base pairs are removed from a DNA segment. Analogously, an insertion represents a mutation where one or more nucleotides are inserted in a particular DNA fragment. Insertions are rare in human genomes and are not relevant for the personalized medicine tests considered in this paper (see paragraph on SNPs below); thus, we ignore them. Also, since sequencing involves alignment to a reference genome, insertions are always detectable. To contend with deletions, we can introduce a special symbol '$-$' in lieu of a deleted letter at a specific location in the genome. This new symbol should be treated as any other base pair, and could potentially exist within a search pattern.

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 members of the same species or paired chromosomes of an individual [49]. The average SNP frequency in the human genome is approximately 1 per 1,000 nucleotide pair. (See [41] for a complete collection of all known SNPs). SNP variations are often associated with how individuals develop diseases and respond to pathogens, chemicals, drugs, vaccines, and other agents, and constitute the main focus of *personalized medicine* testing [12].