2016 IEEE First Conference on Connected Health: Applications, Systems and Engineering Technologies

A Survey of Secure Multiparty Computation Protocols for Privacy Preserving Genetic Tests

Tamara Dugan, M.S.
Department of Computer & Information Science
Indiana University Purdue University Indianapolis
Indianapolis, USA
tmdugan@iu.edu

Xukai Zou, PhD.
Department of Computer & Information Science
Indiana University Purdue University Indianapolis
Indianapolis, USA
xzou@iupui.edu

***Abstract*—We discuss several protocols that apply secure multiparty computation to privacy preserving genetic testing. We categorize methods into those using oblivious finite automata, additive homomorphic encryption, garbled circuits, and private set intersection. Through comparison of performance and security metrics, we aim to make recommendations for efficient and secure multiparty computation protocols for various genetic tests including edit distance, disease susceptibility, identity/paternity/-common ancestry testing, medicine and treatment efficacy for personalized medicine, and genetic compatibility.**

***Index Terms*—smpc; secure multiparty computation; genomics; privacy-preserving.**

I. Introduction

It took 13 years and roughly 2.7 billion dollars to sequence the first human genome [1] and there is promise by 2015 that a human genome can be sequenced in a matter of days for $1000 or less [2]. This reduction in cost has made full genome sequencing more accessible to researchers, government, and even private citizens for use in medical treatment and research, law enforcement, and personal ancestry research. This accessibility also raises privacy concerns since Deoxyribonucleic Acid (DNA) uniquely identifies and contains intimate details about an individual.

Since DNA is the ultimate source of personal information, sharing of this data is particularly dangerous. DNA can reveal information about disease susceptibility and ancestry not only of the individual but of his or her extended family. Disclosure of this information could result in discrimination by insurance companies or reveal unnecessarily intimate information about an individual or even a family member. However, access for medical providers to genetic data can allow clinicians to see if an individual is predisposed to a disease, predict if a patient will respond well to a treatment course, or determine the medication choice that will most effectively be metabolized by the patient to treat an illness or chronic condition. Along with the patient's interest in maintaining the privacy of his or her genomic data, the entity performing the genetic analysis does not want to reveal any details about propriety testing techniques. Both patients and testing agencies need to protect their interests while still being able to work together [3].

Methods like data anonymization have been proposed to secure genomic data, but because of the level of personal information contained in a genome sequence containing 3 billion nucleotides, it is very difficult to anonymize. Many techniques have been tried but have failed to combat re-identification attacks [4], [5].

As an alternative, secure multiparty computation protocols can be used for privacy preserving genetic tests. Secure multiparty computation uses cryptographic protocols to securely compute functions between two parties. This secure function computation can be used by various genetic testing algorithms to ensure that genomic data is protected. This paper describes several such protocols that use secure multiparty computation to both preserve the privacy of the DNA sequence and the intellectual property of the genomic testing agency.

In the following sections, we describe how different secure multiparty computation techniques have been applied to genomic testing protocols. We categorize the protocols into those that implement oblivious finite automata, additive homomorphic encryption, garbled circuits, and private set intersection to maintain genomic privacy. We then identify which secure genomic protocols can be used for computing edit distance, disease susceptibility, identity/paternity/common ancestry testing, personalized medicine, and genetic compatibility and compare them within each subcategory using metrics for performance and security to make a recommendation for the best protocol.

II. Background

A. Genomic Primer

A DNA sequence provides the blueprint for life since each living organism possesses a unique DNA sequence. This unique structure is composed of several smaller units called nucleotides and each nucleotide is composed of one of four bases: adenine (A), cytosine (C), guanine (G), and thymine (T). Protein sequences contain amino acids, sets of three nucleotides, and make up less than 2% of human DNA. A single protein is formed by a sequence of amino-acids and these proteins control different processes within the human body [6].

DNA can be analyzed at a nucleotide level or broken into Short Tandem Repeats (STRs), Single-Nucleotide Polymorphisms (SNPs), or Restricted Fragment Length Polymorphisms (RFLPs). An STR is a small sequence of nucleotides that

978-1-5090-0943-5/16 $25.00 © 2016 IEEE
DOI 10.1109/CHASE.2016.71

repeat several times and the variation in the number of repeats is wide enough among human genomes that it can be used to compare DNA sequences. To determine where the STRs are located, restriction enzymes can be used to cut the DNA into pieces and further analyses can determine the number of repeats [7]. SNPs occur when a single nucleotide differs in the same location between two DNA sequences. Some of these nucleotide differences have been associated with different traits (phenotypes) or disease susceptibilities. An RFLP is created when a DNA sequence is broken into fragments by matching certain subsequences. The lengths of the fragments can be used to analyze the DNA sequences [8].

The edit distance between two strings is the minimum number of insertions, deletions, and substitutions needed to convert string a into string b [9]. Edit distance is a measure of similarity and, in genetics, can be used to see if two DNA sequences are similar. The following dynamic programming algorithm [10] is commonly used to compute edit distance between string a and b where $|a| = n$ and $|b| = m$:

```
D(i,0)= i, 0 <= i <= n
D(0,j) = j, 0 <= j<= m
if a[i] = b[j], then t(i,j) = 0, else t(i,j)=1
D(i,j) = min[D(i-1,j) + 1, D(i,j-1) +1, D(i-1,j-1) + t(i,j)]
```

Smith-Waterman [11] improved upon the protocol of Levenshtein by proposing an algorithm that both computes the edit distance and aligns the two sequences. The primary difference is that the Smith-Waterman protocol allows arbitrary cost for delete, insert, and replace operations instead of a cost of 1.

B. Cryptographic Primer

To facilitate secure multiparty computation, different cryptographic techniques can be used to facilitate the secure exchange of data. Below are some definitions of these cryptographic building blocks that will later be used to construct secure multiparty computation protocols for privacy preserving genetic tests.

1) Oblivious Transfer: In a 1-out-of-k oblivious transfer protocol, k values are sent to a choosing party that has a specific index. The chooser obtains the given value while revealing nothing to the sender [12].

2) Private Information Retrieval: Private Information retrieval is a weaker version of a 1-out-of-k oblivious transfer protocol that could reveal values other than the one corresponding to the index of the chooser [13].

3) Additive Homomorphic Encryption: An additive homomorphic encryption scheme is a public key encryption scheme such that $E(x_1) * E(x_2) = E(x_1 + x_2)$ and $E(x)^c = E(cx)$ [14]. This kind of encryption scheme allows two parties to compute sums of ciphertexts without ever decrypting the values.

III. Classification by Cryptographic Methods

Although some protocols use multiple cryptographic methods to implement secure multiparty computation, we have classified each based on the primary method used. The protocols fall within the following cryptographic categories: Oblivious Finite Automata, Primary Homomorphic Encryption, Garbled Circuits, and Private Set Intersection.

A. Oblivious Finite Automata Methods

A finite automata consists of five elements: a finite set of states, a finite set of input symbols, a transition function, an initial state, and a set of final states. The initial state of the automata consumes the first element of the string and uses the state transition matrix to determine the next state transition. The process continues through the set of states until all elements are consumed and the final state is reached. The string is accepted if all characters are consumed and the last state is an acceptance state [3].

1) Edit Distance: The first method for using oblivious automata to compute edit distance between strands of DNA was proposed by Troncoso-Pastoriza, et al. [3] in 2007. The authors developed a protocol that supported regular expression matching that could handle mutations and sequencing errors. Troncoso-Pastoriza et al.'s protocol involves two parties: an automata definition and a string. The protocol is made up of three sub protocols. The first state transition sub protocol involves a 1-out-of-m oblivious transfer protocol between Alice and Bob where a random blinding value blinds each element with a random state. In the k-th state transition protocol, Alice repeats the first state transition protocol then rotates the rows of the transition matrix a number of columns matching the blinding value from the previous state transition. Bob sends an encrypted binary vector with all zeros except for a 1 corresponding to the state transition chosen from the previous oblivious transfer protocol. Alice takes the encrypted vector and multiplies it by an encrypted version of the state transition matrix to get an encrypted state transition vector since the encryption scheme is additively homomorphic. Alice sends the encrypted vector to Bob and Bob decrypts the vector and chooses the state corresponding to the index of his next character. This process is effectively a 1-out-of-m oblivious transfer through the encrypted vector. For the final sub protocol, Alice creates a binary vector where a position contains a 1 if it is an acceptance state and a 0 otherwise. The elements in the vector are then shifted by a random state number that corresponds to the blinding value used in the previous oblivious transfer with Bob. Alice and Bob perform a 1-out-of-m oblivious transfer where Bob looks up his final state index to see if the value is 0 or 1, reject or accept.

In 2009, Frikken [15] improved upon Troncoso-Pastoriza et al.'s protocol by addressing two performance shortcomings of the prior protocol: linear complexity in the size of the DNA sequence and modular exponentiations on the order of the number of automata states times the number of possible DNA sequence characters. The author's enhanced protocol uses random encoding of character inputs and states to cut down on the number of oblivious transfers. Frikken notes that the number of rounds is reduced from $O(N)$ to $O(1)$ and modular exponentiations are reduced from $O(|Q|*N+N*|\Sigma|)$ to $O(N)$. Additionally, the author benchmarked both protocols

and noted a 2 to 3 order of magnitude speedup in his protocol versus that of Troncoso-Pastoriza et al.

In 2010, Blanton and Aliasgari [16] improved Troncoso-Pastoriza et al.'s protocol in two ways: they adapted the kth state transition sub protocol to skip the encryption of the data before the oblivious transfer and they replaced the oblivious transfer protocol with a more efficient Private Information Retrieval (PIR) protocol to reduce the communication complexity. The authors preserved the security of the transfer protocol by using symmetric PIR. More specifically, the authors flattened the transition matrix into one dimension instead of two, removing the need for the additive homomorphic encryption scheme used by Troncoso-Pastoriza et al. To further improve efficiency, the authors proposed a protocol for multiparty outsourcing of the data computation. In terms of performance, Bob's work drops by a factor of $|\Sigma|+2$ and Alice's work drops by a factor of $\log|Q|$ compared to the Troncoso-Pastoriza et al. protocol. The communication complexity is $O(N|\Sigma||Q|)$ which appears to be more than the previous protocol which had a communication complexity of $O(N(|\Sigma|+|Q|))$. However, the authors note that the security parameter needed for the encryption can end up making up for this difference, meaning the new protocol is at least as efficient.

Sasakawa et al's protocol [17] was quite a bit different than other OFA protocols in that it used non-deterministic finite automata (NFA). Specifically, the authors adapted previous work on approximate matching using oblivious deterministic finite automata (DFA) to non-deterministic finite automata [18]. Because NFA's have considerably fewer states than DFA's in general, oblivious NFA protocols can be more efficient than DFA protocols. The authors propose a protocol between Alice (the automata owner) and Bob (the input string) that involves matrices of private state transitions, public state transitions, and active states. Alice encrypts values of all three matrices using an additive homomorphic encryption scheme and gives the matrices to Bob. For each character of his DNA string, Bob uses a secure multiplication protocol to multiply the encrypted transition matrix entry for his character by the encrypted value of the currently active state. Alice then decrypts the final result and if it is 0, outputs true, false otherwise.

B. Primary Homomorphic Encryption Methods

Though additive homomorphic encryption is used by many secure genomic protocols in conjunction with other cryptographic methods, some protocols avoid the need for data transfer methods like oblivious transfer and private information retrieval all together by using purely additive homomorphic encryption protocols. As previously described in the Cryptographic Primer, additive homomorphic encryption involves use of cryptographic functions that add two encrypted values together without needing to decrypt, compute the sum, and re-encrypt.

1) Edit Distance: Atallah et al. [19] proposed one of the earliest additive homomorphic encryption protocols for secure computation on genomic data. The authors' protocol is rooted in securely executing a dynamic programming algorithm to compute edit distance between two genomic sequences by sharing elements of the dynamic programming algorithm between two parties. To compute the minimum calculation needed by the dynamic programming algorithm, the authors developed a secure protocol for finding the minimum across the split data. Alice has a vector of three of her split values. She encrypts the three values in the vector and sends the vector a' of encrypted values to Bob. Bob generates a vector r with three random values and encrypts each of the random values with Alice's public key. Bob then multiples the corresponding entries within a' and r and permutes the resulting vector before sending to Alice. This operation essentially hides the values in a' by adding a random number. Bob does the same for his split value vector. The parties can then run a minimum value protocol to privately compute the minimum needed by the dynamic programming algorithm. A couple of years later, Atallah and Li [20] provided enhancements to the original protocol and proposed an adaptation of the protocol so that a customer and two agents can split the calculations.

2) Disease susceptibility: Kantarcioglu et al. [21] present a protocol allowing secure count queries on a database of genetic data to be used to compute disease susceptibility. In their protocol, there are three parties: the researcher querying the data, the data storage (DS) center containing encrypted genomic data, and the key holder service (KHS) that issues keys for encrypting and decrypting results. The KHS generates a public key from an additive homomorphic encryption scheme and gives the key to the DS. The DS sends the public key to Bob so he can encrypt his data and send it to the DS. Bob's data consists of a set of SNPs from his genome converted to encrypted binary representations. To initiate the protocol, a researcher sends a query containing the set of SNPs to the DS that he or she wants to match. The DS encrypts the SNPs using the additive homomorphic public key issued by the KHS and computes an algebraic equation with each person's set of encrypted SNPs in the database. If the query matches, the result of the equation will be an encrypted 0, otherwise a random number. The encrypted results for all individuals in the database are sent to the KHS for decryption and if the decrypted result is 0, the KHS increments a counter that will be ultimately sent to the researcher.

Although the details of the server implementation are different, Kantarcioglu et al. and Ayday et al.'s [22] protocols both compare DNA strings using the same mechanism. Four entities are involved in Ayday et al.'s protocol: the patient (P), the Certified Institution (I), the Storage and Processing Unit (SPU), and the Medical Center (MC). The patient provides a sample to the CI for sequencing and the CI encrypts a binary array the size of all possible SNPs with P's public key. The CI sends the encrypted SNP array to the SPU for storage. The patient's private key is randomly split into two shares. One share is given to the SPU and another is given to the MC when it is time to test for disease susceptibility. The MC sends necessary values such as the list of disease related SNPs and the probabilities that the SNPs will suggest the disease being tested to the SPU in cleartext. The SPU uses the patient's

public key to encrypt the values sent by the MC and performs additive homomorphic operations with the encrypted genomic data of the patient to compute a weighted average to determine disease susceptibility. The resulting disease susceptibility is partially decrypted by the SPU and sent to the MC to finish the decryption.

Later, Danezis and De Cristofaro [23] presented two protocols very similar to that of Ayday et al.'s protocol. The authors improved upon the previous protocol by encrypting the SNPs in a different manor that allows them to use a much more efficient additive homomorphic encryption method, Additively Homomorphic Elliptic Curve based El-Gamal, that is both faster and produces smaller ciphertexts. The newer protocols also prevent leaking of which SNPs are being matched and how many are being checked. The authors provide a reference implementation for both Ayday et al's protocol and their own of which the former is adapted to prevent leakage of the number and details of SNPs being checked. In terms of storage, the Danezis and De Cristofaro protocols take only 4.5% of the storage needed by their predecessor and are 4 times faster.

Franz et al. [24] provide a different approach by building on previous work for securely evaluating Hidden Markov Models (HMM) [25], [26]. Specifically, the authors build upon the previous protocols by providing security proofs and analysis of reference implementation performance. In the protocol, Bob uses an additive homomorphic encryption scheme to encrypt his DNA sequence. Alice uses Bob's public key to encrypt probabilities within her HMM. Both parties use optimized versions of secure protocols described in [27] for computing the product and sum of the encrypted values to produce a resulting probability that Bob has the given disease being tested.

3) Identity/Paternity/Ancestry Testing, Personalized Medicine, & Genetic Compatibility: In their paper, Bruekers et al. [7] propose secure protocols for identity testing, common ancestor testing, and paternity testing with one or two parents. The foundation of these protocols is based on comparing the STRs at specific locations on the genome between the parties of interest. The genome comparison algorithm is very similar to Kantarcioglu et al. but is applied to a different set of genetic tests. In the authors' basic protocol to securely test identity, Bob maps each possible allele (sequence of bases) for each possible locus (position of interest) using a random injective function (i.e. collision resistant hash function). Bob then sums up all the hashes for all alleles for all loci and encrypts the result using an additive homomorphic encryption scheme. He sends the encrypted result to Alice and she goes through the same mapping process with her alleles, computes a similar sum, and encrypts the inverse of the result using Bob's public key. Alice then multiplies her and Bob's encrypted values together to compute the difference between their sets of STRs and blinds the result with a random exponent. After Bob decrypts the value from Alice, if the answer is 0, then the two DNA sequences match. The authors adapt the protocol to allow for some error resistance and application to ancestor and paternity testing. The authors provide an example of how the protocol can leak information about the STR profile of Alice to the point that Bob can find out Alice's complete STR profile by running the protocol $N|\Sigma|$ times where N is the number of loci and Σ is the number of possible alleles.

Later De Cristofaro et al. [28] expanded on Bruekers et al. by providing solutions for personalized medicine and genetic compatibility. The authors present a Size-and Position-Hiding Private Substring Match (SPH-PSM) protocol which aims to hide the position markers, the number of markers being tested, and the subset of markers that match if there is not a complete match. The protocol begins by Bob, the genome holder, encrypting each nucleotide in his genome with his public key and sending the results to Alice, the testing lab. Alice encrypts the inverse of each nucleotide in her DNA substring with Bob's public key and multiplies each nucleotide with Bob's, later blinding the result with a random exponent. Alice takes all the results of the individual multiplications and multiplies them together, effectively adding multiple 0's if the strings match. If the strings match exactly, the result decrypted by Bob is 0, otherwise it is a random value.

Djatmiko et al. [29] propose a different approach that only applies to secure personalized medicine. They present a protocol for the Warfarin Dosing Algorithm which uses a linear combination of coefficients with genomic and non-genomic data like height and weight. The protocol involves a client and a server where the client encrypts the genomic data using a public key from an additive homomorphic encryption scheme and the server encrypts the coefficients of the equation using the server's additive homomorphic public key. The server uses PIR to select the subset of encrypted client data that is the genomic data and looks for SNP matches by using additive homomorphic subtraction. The two sides then work together to securely compute the linear equation using additive homomorphic operations.

C. Garbled Circuits

Yao originally proposed the idea of garbled circuits [30], [31]. In his secure two party protocol, a circuit is created with two inputs and one output. Each input and output can accept one bit. In the protocol, Alice creates the circuit and generates 6 keys corresponding to the 6 possible input and output values. Alice sends the key corresponding to her input bit to Bob along with a garbled truth table of four values corresponding to encrypted values of the two possible output keys. Alice sends Bob his two possible input keys using a 1-out-of-2 oblivious transfer protocol. Bob chooses the key corresponding to his input bit and uses his key along with Alice's key to look up the correct value in the garbled truth table and uses both keys to decrypt the output key. Bob then sends the output key to Alice who looks up the output bit mapped to that key.

1) Edit Distance: In one of the earliest secure multiparty genomic protocols using garbled circuits, Jha et al. [32] proposed a method for securely computing edit distance using the Smith-Waterman algorithm. For the basic edit distance

protocol, the authors created 6 Yao garbled circuits to securely implement the standard dynamic program used to compute edit distance. The 6 circuits consisted of 3 circuits to add one, 2 circuits to evaluate the smaller value, and one circuit to execute the equality check needed to determine the $t(i, j)$ value of the dynamic programming algorithm.

Wang et al. [33] build on the Jha et al. protocols for edit distance by reducing the amount of data that must be secured. There are two parties involved, the data consumer (DC) and the data party (DP): a repository of genomic data. The DC runs the query and the DP sanitizes the DNA sequences by replacing SNPs, considered sensitive areas of the genome, with symbols. The two parties then run a protocol that runs a standard dynamic program algorithm for edit distance, i.e. Smith-Waterman. Since only 0.5% of DNA differs between two humans, 99.5% of the DNA sequences can run through a non-secure, faster version of the dynamic program. The other 0.5% that contains the SNPS can be evaluated by using garbled circuits in a secure two party protocol between the DC and DP. The authors create reference implementations of both their protocol for secure multiparty computation and that of Jha et al. and demonstrate a 30 fold improvement in performance [33].

Blanton et al. [34] improve upon Jha et al. and Wang et al.'s protocols in two key ways: reduce the necessary memory from $O(mn)$ to $O(m+n)$ when the edit script, i.e. the actual series of inserts, deletes, and substitutions used to obtain the minimum edit distance, is computed and remove the need for oblivious transfer. The authors present a protocol involving three parties: a client, server 1, and server 2. By cleverly splitting the matrix while running the dynamic program for edit distance, the authors can achieve good computational complexity and reduce the memory needed. The protocol also leverages distributed garbled circuit evaluation between server 1 and 2. To avoid the need for oblivious transfer with the client, the servers use the output wire labels as the input wire labels for the next circuit.

Pu and Liu [35] improve upon Jha et al.'s protocol by implementing similar algorithms for computing edit distance using optimized, highly parallel hardware. Specifically, the authors embed garbling and de-garbling of the circuits into the GPU hardware. Between using techniques from Huang et al. [36] and their hardware optimizations, the authors were able to achieve a speedup of 362.5 times faster than Jha et al.'s fastest protocol for edit distance.

2) Other Tests: Katz and Malka [37] present a secure pattern matching protocol that builds on the work of [38], [39] that can be used for identity testing, paternity testing, and ancestry analysis. In the protocol, Bob creates a database of keywords, i.e. significant DNA subsequences, along with indices where those keywords appear within his sequence. Bob also creates a garbled circuit representing the margin of error of the pattern match and stores the labels with the corresponding values in the database. Alice holds a pattern to match based on the test that Bob requested. Through oblivious pseudorandom function evaluation, Alice learns the circuit input labels that correspond to the indices of her matched pattern in the database. Once Alice has the labels, she evaluates Bob's garbled circuit and Bob decrypts the output to determine if the test was positive. The authors note that the primary performance improvement is that, instead of having a garbled circuit for every possible index that the pattern could match in the DNA sequence, there is only a garbled circuit for the maximum number of matches that could happen. Thus, instead of $O(N * |B|)$ where N is the length of the DNA sequence and $|B|$ the size of the garbled circuit, the time complexity is $O(r * |B|)$ where $r <= N$ and r is the maximum number of repeats expected of a pattern in the DNA sequence which is much smaller than N, in most cases.

Karvelas et al. [40], [41] introduce the use of oblivious random access memory to implement protocols for matching SNPs, DNA fingerprinting through STRs, and disease susceptibility. The basic framework for these protocols involves 3 parties: the person with the DNA sequence, the investigator running the test, and the DNA storage service. The DNA storage service is dividing into two non-colluding servers: the cloud storage server and the proxy server. When an investigator wants to run a test, he accesses all the necessary blocks of data needed to run the test through a custom ORAM protocol based on [42] that hides the data access pattern from the data server so it cannot infer what tests are being run. The investigator applies proxy re-encryption to the whole set of data and then sends it to the data storage service. The cloud storage server and proxy server run a secure two party computation protocol using garbled circuits provided by the investigator. After the computations are complete, the result is sent to the investigator and decrypted.

D. Private Set Intersection

Private set intersection takes a set of inputs X=$x_1, x_2, ...x_m$ from one party and a set of inputs Y=$y_1, y_2, ...y_n$ from another party and securely determines the set of elements in the intersection of X and Y. De Cristofaro et al. provide a protocol [43] that is particularly efficient for private set intersection cardinality (PSI-CA) which only reveals the number of elements that intersect without specifying them directly. In another work by De Cristofaro [44], he and his group propose an efficient authorized private set intersection (APSI) protocol in which client input must be authorized via a certificate authority not involved in the computation of the private set intersection.

Baldi et al. [45] propose three different protocols: one for paternity testing, one for personalized medicine, and one for testing genetic compatibility between potential parents. Each protocol uses a variation on private set intersection to securely compute genomic similarity. The paternity test protocol uses the PSI-CA protocol [43] to quantify the similarity of the genomes of the potential father-child pair. If the cardinality of the set intersection is at or above a certain threshold, then the two individuals are father and child with high probability. The authors note that it takes only 25 markers to get a 99.999% accurate test. The personalized medicine protocol measures genetic markers that determine drug metabolism using an

TABLE I
DNA ELEMENTS USED FOR ANALYSIS

Method	Protocol	SNP	RFLP	STR	Nucleotide
Oblivious Finite Automata	[3]	no	no	no	yes
	[15]	no	no	no	yes
	[16]	no	no	no	yes
	[17]	no	no	no	yes
Primary Homomorphic Encryption	[19], [20]	no	no	no	yes
	[21]	yes	no	no	no
	[7]	no	no	yes	no
	[22]	yes	no	no	no
	[23]	yes	no	no	no
	[28]	yes	no	no	no
	[29]	yes	no	no	no
	[24], [49]	no	no	no	yes
Garbled Circuits	[32]	no	no	no	yes
	[33]	no	no	no	yes
	[35]	no	no	no	yes
	[34]	no	no	no	yes
	[40], [41]	yes	no	yes	no
	[37]	no	no	no	yes
Private Set Intersection	[45]	yes	yes	no	no
	[8]	yes	yes	no	no

APSI protocol [44]. Finally, the genetic compatibility protocol measuring disease risk in offspring was designed using a PSI protocol [46].

De Cristofaro et al. [8] describe protocols for paternity testing, ancestry testing, and personalized medicine using the same PSI methodology as Baldi et al. The primary difference is that the authors use an Android smartphone to store an encrypted version of the genome instead of the cloud. They claim that the smartphone implementation is more portable and potentially more secure since the public key on the phone has a more limited lifetime than if it were stored in the cloud. For the ancestry testing protocol, the authors compute an approximate Jaccard similarity index score [47], a measure of set similarity, on 50 SNPs using a PSI-CA protocol. Instead of analyzing the entire genome, they use a MinHash [48] technique to deterministically sample the genome for the 50 SNPs of interest. A threshold value is chosen for the Jaccard similarity index score above which the two genomic sequence owners are determined to have a common ancestry.

IV. COMPARISON

In the following section, we compare different secure multiparty genomic protocols within like genomic applications. We will analyze level of security, outsourcing capabilities, and performance via computational and communication complexity and elapsed time of reference implementations. Through these comparisons, we aim to provide guidance in choosing the most appropriate protocol for each genomic application.

Genetic tests can be grouped into five categories: edit distance; disease susceptibility; identity, paternity, and common ancestry; personalized medicine; and genetic compatibility. We will examine protocols to determine best practices for each application based on security and performance.

A. Edit Distance

Edit distance is an important metric in genome analysis to measure similarity between genome sequences. It can be used

TABLE II
GENOMIC APPLICATIONS

Method	Protocol	ED	DS	Iden	Anc	Pat	PM	GC
Oblivious Finite Automata	[3]	yes	no	no	no	no	no	no
	[15]	yes	no	no	no	no	no	no
	[16]	yes	no	no	no	no	no	no
	[17]	yes	no	no	no	no	no	no
Primary Homomorphic Encryption	[19], [20]	yes	no	no	no	no	no	no
	[21]	no	yes	no	no	no	no	no
	[7]	no	no	yes	yes	yes	no	no
	[22]	no	yes	no	no	no	no	no
	[23]	no	yes	no	no	no	no	no
	[28]	no	no	yes	yes	yes	yes	yes
	[29]	no	no	no	no	no	yes	no
	[24], [49]	no	yes	no	no	no	no	no
Garbled Circuits	[32]	yes	no	no	no	no	no	no
	[33]	yes	no	no	no	no	no	no
	[35]	yes	no	no	no	no	no	no
	[34]	yes	no	no	no	no	no	no
	[40], [41]	no	yes	yes	yes	yes	yes	yes
	[37]	no	no	yes	yes	yes	yes	yes
Private Set Intersect.	[45]	no	no	no	no	yes	yes	yes
	[8]	no	no	no	yes	yes	yes	no

ED - Edit Distance DS - Disease Susceptibility Iden - Identity Testing Anc - Ancestry Testing Pat - Paternity testing PM - Personalized Medicine GC- Genetic Compatibility.

TABLE III
CRYPTOGRAPHIC METHODS USED BY GENOMIC PROTOCOLS

Method	Protocol	Additive HE	OT	PIR
Oblivious Finite Automata	[3]	Paillier	yes	no
	[15]	no	yes	no
	[16]	no	yes	yes
	[17]	Paillier	yes	no
Primary Homomorphic Encryption	[19], [20]	any	no	no
	[21]	Paillier	no	no
	[7]	AH-El Gamal	no	no
	[22]	BCP	no	no
	[23]	EC-El Gamal	no	no
	[28]	AH-El Gamal, EC-El Gamal	no	no
	[29]	Paillier	yes	yes
	[24], [49]	DGK	no	no
Garbled Circuits	[32]	no	yes	no
	[33]	no	yes	no
	[35]	no	yes	no
	[34]	no	no	no
	[40], [41]	BCP	yes	no
	[37]	no	yes	no
Private Set Intersection	[45]	no	no	no
	[8]	no	no	no

HE - Homomorphic Encryption, OT - Oblivious Transfer,
PIR - Private Information Retrieval

TABLE IV
COMPLEXITY OF GENOMIC PROTOCOLS

Method	Protocol	Comm. Complexity	Comp. Complexity	Ref. Impl.
Oblivious Finite Automata	[3]	$O(N(\|Q\| + \|\Sigma\|))$	$O(N\|\Sigma\|\|Q\|)$	no
	[15]	$O(N(\|Q\|\|\Sigma\|+\|\Sigma\|))$	$O(N\|\Sigma\|\|Q\|)$	yes
	[16]	$O(N\|Q\|\|\Sigma\|)$	$O(N\|\Sigma\|\|Q\|)$	no
	[17]	$O(N\|Q\|^2)$	$O(N\|Q\|^2)$	yes
Primary Homo. Encrypt.	[19], [20]	$O(\|\Sigma\|N)$	$O(\|\Sigma\|N)$	no
	[21]	$O(\alpha k)$	$O(\alpha)$	yes
	[7] (Pat/Anc)	$O(N^{e+1})$	–	no
	[7] (Ident)	$O(N^e)$	–	no
	[22]	–	–	yes
	[23]	–	–	yes
	[28]	$O(N)$	$O(m)$	yes
	[29]	$O(k+d)$	–	partial
	[24], [49]	–	–	yes
Garbled Circuits	[32]	–	–	yes
	[33]	$O(N^3)$	$O(N^3)$	yes
	[35]	–	–	yes
	[34]	$O(\|\Sigma\|N^2)$	$O(\|\Sigma\|N^2)$	no
	[40], [41]	–	–	yes
	[37]	$O(kN)$	$O(uN)$	no
Private Set Intersect.	[45]	–	–	yes
	[8]	–	–	yes

N=length of DNA string $|Q|$=# of states of finite automata $|\Sigma|$=# of distinct characters α=# of encrypted DNA sequences k=security key length L = # of loci d=# retrieved bits e=# marker mismatches m=size of substring u=# of garbled circuits Ref. Impl. = Reference Implementation

TABLE V
BENCHMARK IMPLEMENTATIONS OF GENOMIC PROTOCOLS

Method	Protocol	CPU (GHz)	Cores	RAM (GB)	Bits	# Mk-s/N	Q	On. (s)	Off. (s)
OFA	[3]	–	–	–	–	–	–	–	-
	[15]	2.0	2	0.5	1024	10	10000	8	–
	[16]	–	–	–	–	–	–	–	–
	[17]	2.53	1	0.5	512	150	–	192.3	–
PHE	[19], [20]	–	–	–	–	–	–	–	–
	[21]	3.4	1	2	1024	10	NA	1500	–
	[7]	–	–	–	–	–	–	–	–
	[22]	2.70	1	–	1024	10	NA	10.328	–
	[23]	1.7	1	–	112	1 mil	NA	7	1024
	[28]	3.4	4	16	1024	10	NA	0.0001	414000
	* [29]	2.36	2	32	1024	9	NA	0.3	–
	[24], [49]	2.1	8	4	1024	3 bil	3920	480	–
GC	[32]	3	1	2	NA	25	NA	14	–
	[33]	1.8	2	2	NA	400	NA	28.526	–
	[35]	2	1	–	NA	5000	NA	1520	–
	[34]	–	–	–	–	–	–	–	–
	[40], [41]	2.67	2	32	1024	1	NA	12.39	1209600
	[37]	–	–	–	–	–	–	–	–
PSI	[45] (PM)	2.66	1	–	1024	6	NA	0.00492	12360
	[45] (GC)	2.66	1	–	1024	52	NA	0.01452	4020
	[45] (Pat)	2.66	1	–	1024	25	NA	0.0034	0.0034
	[8] (PM)	1.2	2	1	–	6	NA	0.301	–
	[8] (Anc)	1.2	2	1	–	50	NA	0.394	0.713
	[8] (Pat)	1.2	2	1	–	25	NA	0.399	0.244

N=length of DNA string Q=# of states of finite automata Mks=Markers On. = Online Off. = Offline * partial implementation

to analyze similarities and differences between species and within subsets of the same species. The metric can also be used to align two difference DNA sequences for analysis.

Looking at Table II, we can see there are nine different protocols proposed for edit distance. All are nucleotide based methods, as we see from Table I. We see from Table VI, that Troncoso-Pastoriza et al's protocol, Sasakawa et al's protocol, and Blanton and Aliasgari oblivious finite automata (OFA) protocol all provide proofs of security. Frikken provides a sketch of a proof. All protocols with the exception of Blanton and Aliasgari require a minimum of two parties. However, Blanton et al's garbled circuit (GC) protocol is specifically meant to have three parties so two servers can share data processing. Both of Blanton's protocols and Atallah et al provide a protocol for outsourcing computation of the edit distance to more than one server. Blanton and Aliasgari's OFA protocol is the only protocol that supports outsourcing to more than two parties.

We have previously stated that Frikken's protocol is 2 to 3 orders of magnitude faster than Troncoso-Pastoriza et al. However, in Table IV we see that the communication complexity is slightly higher for Frikken. However, Σ is only 4 in the genomic case so the O complexity is essentially the same. In

TABLE VI
SECURITY ANALYSIS FOR GENOMIC PROTOCOLS

Method	Protocol	Min # of Parties	Maximum Out-sourced Parties	Security Proof
Oblivious Finite Automata	[3]	2	0	yes
	[15]	2	0	sketch
	[16]	2	> 2	yes
	[17]	2	0	yes
Primary Homomorphic Encryption	[19], [20]	2	2	no
	[21]	3	0	yes
	[7]	2	0	no
	[22]	4	0	no
	[23]	4	0	no
	[28]	2	0	yes
	[29]	2	0	no
	[24], [49]	2	0	yes
Garbled Circuits	[32]	2	0	no
	[33]	2	0	no
	[35]	2	0	no
	[34]	3	2	no
	[40], [41]	4	0	sketch
	[37]	2	0	yes
Private Set Intersection	[45]	2	0	no
	[8]	2	0	no

All protocols assume semi-honest security model

Table IV, it also appears that Sasakawa et al's protocol has a greater communication and computational complexity than Troncoso-Pastoriza et al. even though their protocol should be an improvement. Since the protocol uses NFA's instead of DFA's, it actually has an exponentially smaller number of states than Troncoso-Pastoriza et al.'s protocol, meaning the communication and computational complexities are less and it is in fact an improvement. Additionally, we have noted previously that Blanton and Aliasgari reduces Bob's work by a factor of $|\Sigma| + 2$ and Alice's work by a factor of $\log |Q|$ compared to the Troncoso-Pastoriza et al. protocol. If we look at Table IV, it appears that this computational complexity improvement is at the expense to the communication complexity. As we have previously noted, the authors state that the security parameter needed for the encryption absorbs the difference in complexities so they end up being the same.

In terms of performance, if we examine computational complexity, normalize reference implementation times, and consider ability to outsource from Tables IV, V, and VI, our order of performance from best to worst for these methods would be: Atallah et al, Sasakawa et al, Blanton and Aliasgari (OFA), Frikken, Troncoso-Pastoriza et al, Blanton et al (GC), Wang et al, Pu et al, and Jha et al. Atallah et al. has the best computational complexity and has support for outsourcing to two parties. Sasakawa et al. has an exponentially smaller number of states compared to other oblivious automata protocols. Blanton et al (OFA) and Frikken have similar computational complexities but Blanton and Aliasgari supports outsourcing. Troncoso-Pastoriza et al. has similar computational complexity to Blanton and Aliasgari (OFA) and Frikken, but Frikken notes that his protocol is 100 to 1000 times faster [15]. Next in computational complexity is Blanton et al (GC). If we consider computational complexity and normalize implementation times, then next is Wang et al, followed by Pu et al, followed by Jha et al. Overall, in terms of performance and security, Sasakawa et al is the best but, for pure performance, Atallah et al. wins.

B. Disease Susceptibility

Disease susceptibility involves determining the probability of developing a certain disease. When related to genomics, it involves using associations of certain genes with certain diseases to compute a probability of disease. There are two approaches to this: the typical approach is to use SNP association to compute a probability of disease; the other method is to pass the entire genome sequence through a HMM to compute the disease probability.

As we see from Table II, there are 5 secure disease susceptibility calculation protocols. The majority of the protocols are marker based with the exception of Franz et al. (Table I). Table VI illustrates that Kantarcioglu et al. and Franz et al. provide proofs of security, with Karvelas et al. providing only a sketch. We can see that Franz et al. require the minimum number of parties which reduces possible sources of security leaks. However, there is also a benefit to requiring more parties because it would take more collusion to breach security.

In terms of performance, Kantarcioglu et al. provide the only direct analysis of complexity bounds. In terms of elapsed time, to normalize the data between nucleotide and marker analysis, we normalize to the size of the entire genome. Danezis and De Cristofaro [23] note that it takes 1 million SNPs to cover the entire genome. Based on this normalization, Danezis and De Cristofaro is by far the fastest at 7 seconds. Franz et al is 8 minutes, followed by Ayday et al. (30 ms per marker [22]) and Kantarcioglu et al (assuming 1 record in the database [21]) both at 8.33 hrs and Karvelas et al at a whopping 3441.67 hrs. Franz et al. have more cores than the other implementations but the slower protocols are more than 8 times slower. If guarantee of security is key, Franz et al is the desired protocol. If real time query speed is necessary, Danezis and De Cristofaro should be used.

C. Identity/Paternity/Ancestry Testing

Identity testing involves analyzing two samples of DNA to see if there are certain matches to indicate that the sample came from the same person. Paternity testing aims to analyze two samples to see if there is enough similarity in specific sections of the genome to indicate that one sample comes from the parent and the other sample comes from the biologically child. Ancestry testing is a broader test that examines the genome of two individuals to see if they share common biological ancestry beyond a parent/child relationship.

In Table II, we see there are 6 protocols that implement secure paternity testing. From Table I, we see that all paternity testing protocols but Katz and Malka are marker based. Karvelas et al. is the only protocol in this set that requires more than 2 minimum parties to run the protocol. In terms of security, we see from Table VI that Katz and Malka and De Cristofaro et al's additive homomorphic encryption (AHE) protocol provide proofs of security. Karvelas et al provides only a sketch. In terms of performance, there are reference implementations for 4 out of the 6 protocols. Bruekers et al and Katz and Malka do not provide a reference implementation. Of the two, only Katz and Malka provide a computational complexity, which is linear in the size of the DNA sequence, indicating slow performance if the entire genome is processed. If we normalize the timings of the other protocols for 25 markers, which provides 99.999% accuracy [45], we see that De Cristofaro et al's AHE protocol takes 0.00025 seconds, Baldi et al. takes 0.0034 seconds, De Cristofaro et al's PSI protocol takes 0.399 seconds, and Karvelas et al's takes 309.75 seconds. De Cristofaro et al's AHE protocol uses slightly faster hardware but not enough to account for the difference between its timing and Baldi et al.'s. Baldi's offline speed is considerably faster at 0.0034 seconds versus that of De Cristofaro et al's AHE protocol that requires 115 hours. Since De Cristofaro et al's AHE protocol is both provably secure and has the fastest online speed, it is the best protocol if offline time is not an issue. If both online and offline time must be fast, Baldi et al's protocol is ideal.

All of the authors of the paternity testing protocols, with the exception of Baldi et al., present ancestry testing protocols. Most of the remaining 5 protocols, use the paternity testing

protocol and apply it to ancestry testing with the exception of De Cristofaro et al's PSI based protocol. If we normalize for 50 SNPs, the number needed for ancestry testing [8], we again see De Cristofaro et al's AHE protocol is the fastest at 0.0005 seconds, followed by De Cristofaro et al's PSI protocol at 0.394 seconds, then Karvelas et al 619.5 seconds, and finally Katz and Malka's protocol. Again, De Cristofaro et al's AHE protocol is the best in both security and speed. If both online and offline performance must be fast, De Cristofaro et al's PSI protocol would be used.

Identity protocols are provided by 4 out of the 6 paternity test authors, the exceptions being Baldi et al. and De Cristofaro et al.'s PSI protocol. De Cristofaro et al's AHE protocol is the best in both security and speed for identity testing as well.

Overall, the De Cristofaro et al AHE protocol is the best for identity, paternity, or ancestry testing.

D. Personalized Medicine

Personalized medicine focuses on using genetic data to inform medical treatment. In particular, certain genetic makeups metabolize certain drugs better than others. Personalized medicine aims to create a custom drug regimen based on an individual's genetic makeup.

As we see in Table II, there are 6 authors that describe protocols that can be applied to personalized medicine. From Table I, we see that all but Katz and Malka are marker based protocols. In terms of security, Table VI illustrates that Katz and Malka and De Cristofaro et al provide a formal proof of security while Karvelas et al just provide a sketch. All but Karvelas et al. require a minimum of two parties.

In terms of performance, all of the authors provide benchmark values for reference implementations, except for Katz and Malka. If we normalize online times to 50 SNPs in Table V, we first see that Karvelas et al. is considerably slower than the other protocols at 619.5 seconds for 50 SNPs. De Cristofaro et al. AHE is the fastest at 0.0005 seconds followed by Baldi et al. at 0.041 seconds and Djatmiko et al and De Cristofaro et al. PSI at 1.67 s and 2.5 s, respectively. Note that Djatmiko et al only provide a partial implementation. Karvelas et al is considerably slower followed by Katz and Malka. Baldi et al has the best offline performance.

De Cristofaro et al.'s AHE protocol provides the best performance and security. If offline speed is an issue, Baldi et al provides better offline and comparable online speed.

E. Genetic Compatibility

Certain couples with known risk factors for diseases with a strong genetic component choose to pursue genetic counseling. As part of this counseling, a series of tests comparing the couples' genomes are run to assess their genetic (in)compatibility.

In Table II, we see that 4 protocols can be applied to personalized medicine. Karvelas et al's and De Cristofaro et al's AHE description of pattern matching of SNP's can be applied to searching for genetic markers indicating disease risk. The resulting disease risk from the two partners can be evaluated to compute the total risk for their future children. All but Katz and Malka are marker based protocols. According to Table VI, Katz and Malka and De Cristofaro et al provide a formal proof of security while Karvelas et al just provide a sketch. All but Karvelas et al require a minimum of two parties.

As before, if we normalize each protocol for a given number of SNPs, i.e. 52, De Cristofaro et al's AHE protocol is the fastest at 0.00052 seconds, followed by Baldi et al's protocol at 0.01452 seconds and lastly Karvelas et al at 644.28 seconds and finally Katz and Malka.

As with personalized medicine, De Cristofaro et al.'s AHE protocol is the fastest and most secure, but if offline speed must be faster Baldi et al is best.

V. Observation and recommendation

From previous Tables (I to VI), we observe De Cristofaro et al's AHE protocol dominates tests that involve basic marker matching. If offline time must be faster, the PSI protocols provide a less secure alternative. Additive homomorphic encryption protocols dominate disease susceptibility as well but provable security is somewhat costly in terms of speed. Edit distance is a bit trickier. When the whole genome is involved, it is very difficult to quickly and securely compare two 3 billion length DNA strings. The marker based protocols have a distinct advantage in terms of speed but Saskawa et al and Atallah et al provide protocols that are usable for edit distance.

TABLE VII
Recommended Protocols By Genetic Test

Genetic Test	PS	ONP	OFFP
Edit Distance	[17]	[19], [20]	–
Disease Susceptibility	[24], [49]	[23]	[23]
Paternity Testing	[28]	[28]	[45]
Ancestry Testing	[28]	[28]	[8]
Identity Testing	[28]	[28]	[28]
Personalized Medicine	[28]	[28]	[45]
Genetic Compatibility	[28]	[28]	[45]

PS - Provable Security; ONP - Online Performance; OFFP - Offline Performance

Based on these observations, Table VII provides a summary of recommended protocols by genetic test. De Cristofaro et al's AHE protocol is the fastest and most secure protocol for identity testing, paternity testing, ancestry testing, personalized medicine, and genetic compatibility. It is also commercially viable since all of the applications of this protocol have subsecond timings. Because of the additive homomorphic encryption piece, offline time is slow but can be done in pieces if needed. In terms of disease susceptibility, Franz et al. provide a fast and secure protocol. Danezis and De Cristofaro's protocol is faster but the small tradeoff in speed of Franz et al's protocol is justifiable for proven security. All of these applications have viable real time solutions.

Unlike the previously mentioned tests, edit distance protocols are currently too slow to be executed in real time on the full genome. Since edit distance requires access to an entire portion of the genome and not just marker data, it requires especially sensitive data. Because of this, we recommend Sasakawa et al. since it is provably secure.

VI. Conclusion

With the exception of edit distance, there are fast and secure multiparty computation protocols for genetic testing that are commercially viable. However, all is not lost in terms of edit distance. The best edit distance protocols are usable on reasonably small subsequences of the genome.

Since the marker based methods have been proven to be fast and secure, work should be done to implement these protocols between encrypted cloud storage providers holding DNA and genetic testing facilities. By using these protocols, we can alleviate many of the fears associated with misuse of genomic data and accelerate the advances in genomics that continue to revolutionize modern medicine.

References

[1] ornl.gov/hgmis, "Human genome project," Apr. 2015. [Online]. Available: http://web.ornl.gov/sci/techresources/Human_Genome/index.shtml

[2] E. C. Hayden, "Is the $1,000 genome for real?" Apr. 2015. [Online]. Available: http://www.nature.com/news/is-the-1-000-genome-for-real-1.14530

[3] J. R. Troncoso-Pastoriza, S. Katzenbeisser, and M. Celik, "Privacy preserving error resilient dna searching through oblivious automata," in *ACM CCS'07*. ACM, 2007, pp. 519–528.

[4] M. Gymrek, A. L. McGuire, D. Golan, E. Halperin, and Y. Erlich, "Identifying personal genomes by surname inference," vol. 339, no. 6117, pp. 321–324, 2013.

[5] M. Humbert, E. Ayday, J.-P. Hubaux, and A. Telenti, "Addressing the concerns of the lacks family: Quantification of kin genomic privacy," in *ACM CCS'13*, 2013, pp. 1141–1152.

[6] wikipedia.org, "Dna," Mar. 2015. [Online]. Available: http://en.wikipedia.org/wiki/DNA

[7] F. Bruekers, S. Katzenbeisser, K. Kursawe, and P. Tuyls, "Privacy-preserving matching of dna profiles." *IACR Cryptology ePrint Archive*, vol. 2008, p. 203, 2008.

[8] E. De Cristofaro, S. Faber, P. Gasti, and G. Tsudik, "Genodroid: are privacy-preserving genomic tests ready for prime time?" in *Proceedings of the 2012 ACM workshop on Privacy in the electronic society*. ACM, 2012, pp. 97–108.

[9] V. I. Levenshtein, "Binary Codes Capable of Correcting Deletions, Insertions and Reversals," *Soviet Physics Doklady*, vol. 10, p. 707, Feb. 1966.

[10] D. Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. New York, NY, USA: Cambridge University Press, 1997.

[11] "Identification of common molecular subsequences," *Journal of Molecular Biology*, vol. 147, no. 1, pp. 195 – 197, 1981.

[12] M. Naor and B. Pinkas, "Efficient oblivious transfer protocols," in *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms*, ser. SODA '01, 2001, pp. 448–457.

[13] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan, "Private information retrieval," *J. ACM*, vol. 45, no. 6, pp. 965–981, Nov. 1998.

[14] "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology — EUROCRYPT '99*, ser. Lecture Notes in Computer Science, J. Stern, Ed., 1999, vol. 1592.

[15] K. B. Frikken, "Practical private dna string searching and matching through efficient oblivious automata evaluation," in *Data and Applications Security XXIII*. Springer, 2009, pp. 81–94.

[16] M. Blanton and M. Aliasgari, "Secure outsourcing of dna searching via finite automata," in *Data and Applications Security and Privacy XXIV*. Springer, 2010, pp. 49–64.

[17] H. Sasakawa, H. Harada, D. duVerle, H. Arimura, K. Tsuda, and J. Sakuma, "Oblivious evaluation of non-deterministic finite automata with application to privacy-preserving virus genome detection," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*. ACM, 2014, pp. 21–30.

[18] "Finding approximate patterns in strings," *Journal of Algorithms*, vol. 6, no. 1, pp. 132 – 137, 1985.

[19] M. J. Atallah, F. Kerschbaum, and W. Du, "Secure and private sequence comparisons," in *Proceedings of the 2003 ACM workshop on Privacy in the electronic society*. ACM, 2003, pp. 39–44.

[20] M. J. Atallah and J. Li, "Secure outsourcing of sequence comparisons," *Inter. J. of Info. Security*, vol. 4, no. 4, pp. 277–287, 2005.

[21] M. Kantarcioglu, W. Jiang, Y. Liu, and B. Malin, "A cryptographic approach to securely share and query genomic sequences," *Info. Tech. in Biomedicine, IEEE Trans. on*, vol. 12, no. 5, pp. 606–617, 2008.

[22] E. Ayday, J. L. Raisaro, and J.-P. Hubaux, "Privacy-enhancing technologies for medical tests using genomic data," in *NDSS'13*, 2013, pp. –1–1.

[23] G. Danezis and E. De Cristofaro, "Fast and private genomic testing for disease susceptibility," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*. ACM, 2014, pp. 31–34.

[24] M. Franz, B. Deiseroth, K. Hamacher, S. Jha, S. Katzenbeisser, and H. Schröder, "Towards secure bioinformatics services (short paper)," in *Financial Cryptography and Data Security*. Springer, 2012, pp. 276–283.

[25] "Private predictions on hidden markov models," *Artificial Intelligence Review*, vol. 34, no. 1, 2010.

[26] P. Smaragdis and M. Shashanka, "A framework for secure speech recognition," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 15, no. 4, pp. 1404–1413, May 2007.

[27] B. Deiseroth, M. Franz, K. Hamacher, S. Jha, S. Katzenbeisser, and H. Schröder, "Secure computations on real-valued signals," 2010.

[28] E. De Cristofaro, S. Faber, and G. Tsudik, "Secure genomic testing with size-and position-hiding private substring matching," in *12th ACM workshop on pri. in elec. society*, 2013, pp. 107–118.

[29] M. Djatmiko, A. Friedman, R. Boreli, F. Lawrence, B. Thorne, and S. Hardy, "Secure evaluation protocol for personalized medicine," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*. ACM, 2014, pp. 159–162.

[30] A. C.-C. Yao, "How to generate and exchange secrets," in *Foun. of Comp. Sci., 27th Annu. Symp. on*, Oct 1986, pp. 162–167.

[31] "A proof of security of yao's protocol for two-party computation," *Journal of Cryptology*, vol. 22, no. 2, 2009.

[32] S. Jha, L. Kruger, and V. Shmatikov, "Towards practical privacy for genomic computation," in *Security and Privacy, 2008. SP 2008. IEEE Symposium on*. IEEE, 2008, pp. 216–230.

[33] R. Wang, X. Wang, Z. Li, H. Tang, M. K. Reiter, and Z. Dong, "Privacy-preserving genomic computation through program specialization," in *ACM CCS'09*. ACM, 2009, pp. 338–347.

[34] M. Blanton, M. J. Atallah, K. B. Frikken, and Q. Malluhi, "Secure and efficient outsourcing of sequence comparisons," in *Computer Security–ESORICS 2012*. Springer, 2012, pp. 505–522.

[35] S. Pu and J.-C. Liu, "Computing privacy-preserving edit distance and smith-waterman problems on the gpu architecture." *IACR Cryptology ePrint Archive*, vol. 2013, p. 204, 2013.

[36] Y. Huang, D. Evans, J. Katz, and L. Malka, "Faster secure two-party computation using garbled circuits," in *Proceedings of the 20th USENIX Conf. on Security*, ser. SEC'11, 2011, pp. 35–35.

[37] J. Katz and L. Malka, "Secure text processing with applications to private dna matching," in *ACM CCS'10*. ACM, 2010, pp. 485–492.

[38] "Keyword search and oblivious pseudorandom functions," in *Theory of Cryptography*, ser. LNCS, J. Kilian, Ed., 2005, vol. 3378.

[39] "Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries," in *Theory of Cryptography*, ser. LNCS, R. Canetti, Ed., 2008, vol. 4948.

[40] N. Karvelas, A. Peter, S. Katzenbeisser, E. Tews, and K. Hamacher, "Privacy-preserving whole genome sequence processing through proxy-aided oram," in *Proceedings of the 13th Workshop on Privacy in the Electronic Society*. ACM, 2014, pp. 1–10.

[41] N. P. Karvelas, A. Peter, S. Katzenbeisser, and S. Biedermann, "Efficient privacy-preserving big data processing through proxy-assisted oram." *IACR Cryptology ePrint Archive*, vol. 2014, p. 72, 2014.

[42] P. Williams, R. Sion, and A. Tomescu, "Privatefs: A parallel oblivious file system," in *ACM CCS'12*.

[43] E. D. Cristofaro, P. Gasti, and G. Tsudik, "Fast and private computation of cardinality of set intersection and union," Cryptology ePrint Archive, Report 2011/141, 2011, http://eprint.iacr.org/.

[44] "Linear-complexity private set intersection protocols secure in malicious model," in *Advances in Cryptology - ASIACRYPT 2010*, ser. LNCS, M. Abe, Ed., 2010, vol. 6477.

[45] P. Baldi, R. Baronio, E. De Cristofaro, P. Gasti, and G. Tsudik, "Countering gattaca: efficient and secure testing of fully-sequenced human genomes," in *ACM CCS'11*.

[46] "Fast secure computation of set intersection," in *Security and Cryptography for Networks*, ser. LNCS, J. Garay and R. De Prisco, Eds., 2010, vol. 6280.

[47] P. Jaccard, "Étude comparative de la distribution florale dans une portion des Alpes et des Jura," *Bulletin del la Société Vaudoise des Sciences Naturelles*, vol. 37, pp. 547–579, 1901.

[48] A. Broder, "On the resemblance and containment of documents," in *Compression and Complexity of Sequences 1997. Proceedings*, Jun 1997, pp. 21–29.

[49] M. Franz, B. Deiseroth, K. Hamacher, S. Jha, S. Katzenbeisser, and H. SchröDer, "Secure computations on non-integer values with applications to privacy-preserving sequence analysis," *Inf. Secur. Tech. Rep.*, vol. 17, no. 3, pp. 117–128, Feb. 2013.