# GenoDroid: Are Privacy-Preserving Genomic Tests Ready for Prime Time?

Emiliano De Cristofaro[1] Sky Faber[2] Paolo Gasti[3,*] Gene Tsudik[2]

[1] Palo Alto Research Center
Palo Alto, CA
Emiliano.DeCristofaro@parc.com

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

[3] New York Institute of Technology
New York, NY
pgasti@nyit.edu

## ABSTRACT

As fast and accurate sequencing of human genomes becomes affordable, it is expected that individuals will soon be able to carry around copies of their sequenced DNA, using it for medical, identification, and social purposes. This will undoubtedly prompt a wide range of new and interesting genomic applications. However, the very same progress raises some worrisome privacy issues, since a genome represents a treasure trove of highly personal and sensitive information. Some recent research explored privacy-preserving personal genomic operations by applying (or customizing) cryptographic protocols based on techniques such as: conditional oblivious transfer, garbled circuits, and homomorphic encryption. In this paper, we take this line of work a step further by investigating real-world practicality and usability of (as well as interest in) some of these methods. Motivated by both medical and social applications, we aim to test viability of privacy-agile computational genomic tests in a portable and pervasive setting of modern smartphones. We design a personal genomic toolkit (called GenoDroid), implement it on the Android platform, assess its performance, and conduct a pilot usability study that yields some interesting results.

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

**General Terms:** Security.

**Keywords:** Privacy, DNA, Cryptographic Protocols.

## 1. INTRODUCTION

During the last several decades, the scientific community made significant efforts to improve accuracy, and reduce the cost of, Full Genome Sequencing (FGS), making prices drop significantly faster than Moore's law would otherwise predict [56, 65]. (See, for instance, the $3B, 13-year Human Genome Project [41] and [44, 66].)

A genome represents the entirety of a specific organism's biological information. The availability of *fully sequenced* – and not only human – genomes naturally opens up new and exciting frontiers in numerous fields, including bioinformatics, genomics, genetics, and medicine. In particular, the vision of personalized medicine has been one of the driving forces behind FGS research. Its goal is a set of genomic tests that assess individuals' risk for major diseases, such as diabetes and cancer, as well as at targeted screening and preemptive intervention [20]. Indeed, genetic information already guides doctors toward accurate diagnosis and treatment. However, while some diseases (e.g., Huntington's) are caused by mutations in a single gene and are easily tested *in vitro*, the risk of developing other diseases depends on multiple genes, which makes them difficult to identify. Low-cost genetic sequencing provides researchers with much more genomic information, and enables them to identify new genetic variations as well as run more complicated tests.

Full genome sequencing also facilitates other new applications, such as as paternity, ancestry and genetic compatibility testing. Although less critical than personalized medicine, these more "social" applications are no less exciting, partly because of their (expected) broader appeal. Current lab-based, physical versions of these tests are both time-consuming and privacy-invasive. Performing them computationally makes them much more enticing and accessible.

More generally, we believe that in not-too-distant future, numerous genomic tests and operations will no longer be performed *in vitro*, but *in silico*, i.e., using digitized genomes and specialized computational techniques [37], possibly without the involvement of third-party testing facilities.

### 1.1 Motivation

Despite numerous benefits of low-cost FGS, a number of serious ethical and privacy concerns have emerged [23, 24, 69]. Besides uniquely identifying its owner, a fully sequenced human genome contains information about one's ethnic heritage, phenotypic traits, and predisposition to numerous diseases and conditions, including mental disorders [31, 17, 33]. A virtual *treasure trove* of frighteningly personal and sensitive information is contained in one's genome. Traditional approaches to health care privacy, such as de-identification or aggregation [49, 38], are not helpful in this context, since the genome is the ultimate identifier [50, 51]. A recent study [64] shows that a person's DNA could even be inferred from RNA data (often published in research repositories) even though it was previously assumed not to yield any information about its owner.

Consequently, in order for computational genetic tests on fully sequenced genomes to become accepted and commonplace, efficient and privacy-preserving versions of such tests need to be developed. This poses a number of challenges:

1. ***Privacy***: Given its extreme sensitivity, an individual should

[*] Work done while at University of California, Irvine

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*WPES'12,* October 15, 2012, Raleigh, North Carolina, USA.
Copyright 2012 ACM 978-1-4503-1663-7/12/10 ...$10.00.

ideally never disclose personal genomic information. However, one should be able to allow others (e.g., individuals, doctors, or researchers) to run specific genetic tests that yield nothing beyond their intended results.

2. ***Accuracy***: Computational genomic tests should guarantee accuracy and reliability comparable to current (and widely accepted) lab-based *in-vitro* equivalents. For example, a software implementation of the paternity test on fully sequenced genomes should offer at least the same confidence as its *in-vitro* counterpart, currently admissible in a court of law.

3. ***Efficiency***: Computational genomic tests should incur minimal storage, communication, and computational costs, while satisfying privacy requirements associated with a given test type.

4. ***Portability and Accessibility***: Since a genome is arguably the most sensitive type of personal information, how and where should a user's genome (that contains about $3 \cdot 10^9$ letters) be stored? In the cloud? On a home PC? In a physician's office? On a smartphone? At the health insurance site? A closely related issue is: how should genomes be accessed?

5. ***Usability***: Computational genomic tests should be usable by, and meaningful to, regular non-tech-savvy users. This translates into non-trivial questions, such as: how much understanding should be expected from a user running a test? What information (and at what level of granularity) should be presented to the user as part of a test and as its outcome?

## 1.2 Goals and Outline

Although widespread and affordable availability of fully sequenced human genomes makes it increasingly appealing to perform computational genetic tests, it also raises concerns in terms of simultaneously guaranteeing security, privacy and efficiency. The security research community has been attuned to the emergence of full genome sequencing and a few specialized privacy-preserving cryptographic techniques have been proposed in recent literature. (See Section 8 for a discussion of related work). However, to the best of our knowledge, practicality and usability of such techniques have not been assessed thus far. This is the main goal of this paper.

By carefully designing privacy-preserving mechanisms that emulate *in-vitro*, highly accurate tests, our work demonstrates that secure computational genomic tests are viable today. We present a framework and an implemented toolkit, called *GenoDroid*. It incorporates several techniques offering efficient privacy-preserving genomic testing that meets most aforementioned challenges. In order to demonstrate ubiquity, GenoDroid runs on commodity Android smartphones (though it is not limited to this platform). We also conducted a pilot user study to explore usability and acceptability of proposed techniques.

We focus on the following tests:

- **RFLP- and SNP-based Paternity Tests** establish whether or not a male individual is the biological father of another individual, using genetic fingerprinting based on either Restriction Fragment Length Polymorphisms (RFLP) or Single-Nucleotide Polymorphism (SNP).
- **Ancestry and Genealogical Testing** allows individuals to trace their lineage by analyzing their genomic information. The scope of such tests is often quite heterogeneous. Ancestry testing is useful in a myriad of health-related applications (e.g., susceptibility to diseases common to certain populations). It is also increasingly used in social or recreational scenario, e.g., to map one own genetic heritage or find common ancestry.
- **Personalized Medicine (PM) Testing** provides a significant paradigm shift in health care, aiming at a more precise and powerful type of medicine [74], where diagnosis, treatment, and medication is tailored to the precise genetic makeup of the individual patient. For example, the US Food and Drug Administration (FDA) already recommends testing for mutations in the thiopurine S-methyltransferase (*tpmt*) gene, prior to prescribing 6-mercaptopurine and azathioprine – two drugs used for treating childhood leukemia and autoimmune diseases [4].

Due to extreme sensitivity of human genomic material, for each considered test, we design and implement a privacy-preserving protocol that securely realizes the corresponding computation. Our protocols only yield the test results and do not disclose individuals' genomic information. Furthermore, if the the nature of the test involves sensitive information (e.g., it is a trade secret or is covered by a patent), the contents of the test are also concealed.

**Organization:** The rest of the paper is structured as follows: Section 2 presents a genomics primer and a few cryptographic building blocks, while Section 3 introduces the GenoDroid framework. Then, Section 4, 5, and 6 present privacy-preserving paternity, ancestry, and personalized medicine testing, respectively. Section 7 presents a usability assessment of one of our smartphone applications and Section 8 surveys related work. Finally, we conclude in Section 9.

# 2. BACKGROUND

This section provides background information on genomics, our notation, and cryptographic tools used in the rest of the paper. (It can be skipped with no loss of continuity.)

## 2.1 Genomics Primer

Genomes represent the entirety of an organism's hereditary information. In humans, the genome is encoded in double-stranded DeoxyriboNucleic Acid (DNA) molecules, consisting of two long complementary polymer chains of four simple units called nucleotides, represented by the letters A, C, G, and T. The human genome comprises (about) 3 billion nucleotides. The first genomes to be sequenced were those of a virus and a mitochondrion in late 70's [62]. Since then, *Genomics* has made exceptional progress toward understanding the genome – for more information, we refer to [12]. We now overview some key concepts used in the rest of the paper.

**Restriction Fragment Length Polymorphisms (RFLPs)** refers to a difference between samples of homologous DNA molecules that come from differing locations of restriction enzyme sites, and to a related laboratory technique by which these segments can be illustrated. In RFLP analysis, the DNA sample is broken into pieces (digested) by restriction enzymes and the resulting restriction fragments are separated according to their lengths by gel electrophoresis. Thus, in short, RFLP provides information about the length (and not the composition) of the DNA sub-sequences occurring between known sub-sequences that are recognized by particular enzymes. Although it is being progressively superseded by inexpensive DNA sequencing technologies, RFLP analysis was the first DNA profiling technique inexpensive enough to see widespread application and is still in use today. RFLP probes are frequently used in genome mapping and in variation analysis, such genotyping, forensics, paternity tests, hereditary disease diagnostics. (For more details, see [55].)

**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 [68]. The average SNP frequency in the human genome is approximately 1 per 1,000 nucleotide pair. (See [54] 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. Thus SNPs are key enablers in realizing *personalized medicine* [18]. Moreover, they are used in genetic disease and disorder testing, as well as to compare genome regions between cohorts in Genome-Wide Association Studies (GWAS) [14].

## 2.2 Notation

In the rest of this paper, we denote a digital copy of an individual's fully sequenced genome by $\mathcal{G} = \{(b_1||1), \ldots, (b_n||n)\}$, where $b_i \in \{\text{A, G, C, T, –}\}$, $n$ is the genome length, and "$||$" denotes concatenation. The "–" symbol is needed to handle DNA mutations corresponding to *deletion*, i.e., where a portion of a chromosome is missing [48]. It is also used when the sequencing process fails to determine a nucleotide. In case of *insertion* mutation in the genome, e.g., an 'A' is added between positions $x$ and $x+1$, we add $(A||x||1)$. Similarly, if insertion involves multiple nucleotides. Since insertions are rare in human genomes (in the order of $0.1\%$), we do not consider them in this paper. We use the $|str|$ to denote the length of string $str$ and $|A|$ to denote the cardinality of set $A$. Finally, $r \leftarrow G$ indicates that $r$ is chosen uniformly at random from set $G$.

## 2.3 Cryptography

We now overview a set of cryptographic notions and tools used in the rest of the paper. For ease of exposition, we omit the basics and refer to [45, 52] for other notions, such as encryption and signature schemes, hash functions, and number-theoretic assumptions.

**Private Set Intersection Cardinality (PSI-CA)** [32]: a protocol between server with input $\mathcal{S} = \{s_1, \ldots, s_w\}$, and client with input $\mathcal{C} = \{c_1, \ldots, c_v\}$. At the end of the protocol, client learns $|\mathcal{S} \cap \mathcal{C}|$. PSI-CA securely implements: $\mathcal{F}_{\mathsf{PSI\text{-}CA}} : (\mathcal{S}, \mathcal{C}) \mapsto (\perp, |\mathcal{S} \cap \mathcal{C}|)$.

**Authorized Private Set Intersection (APSI)** [28]: a protocol between server with input $\mathcal{S} = \{s_1, \ldots, s_w\}$, and client with input $\mathcal{C} = \{c_1, \ldots, c_v\}$ and $\mathcal{C}_\sigma = \{\sigma_1, \ldots, \sigma_v\}$, where $\sigma_i = \mathsf{Sig}(sk_{\mathsf{CA}}, c_i)$, for $i = 1, \ldots, v$, and authorization authority CA. At the end of the protocol, client learns: $\mathsf{ASI} \overset{\text{def}}{=} \mathcal{S} \cap \{c_i \mid \mathsf{Ver}(pk_{\mathsf{CA}}, \sigma_i, c_i) = 1\}$. APSI securely implements: $\mathcal{F}_{\mathsf{APSI}} : (\mathcal{S}, (\mathcal{C}, \mathcal{C}_\sigma)) \mapsto (\perp, \mathsf{ASI})$.

**Secure Hamming Distance (SHD)**: a protocol between server with input string $\mathtt{S}$, and client with input string $\mathtt{C}$ such that $|\mathtt{S}| = |\mathtt{C}|$ At the end of the protocol, client learns $\mathsf{HD}(\mathtt{S}, \mathtt{C})$ (where $\mathsf{HD}$ denotes Hamming Distance, i.e., the number of positions at which the corresponding symbols are different). SHD securely implements: $\mathcal{F}_{\mathsf{SHD}} : (\mathtt{S}, \mathtt{C}) \mapsto (\perp, \mathsf{HD}(\mathtt{S}, \mathtt{C}))$.

**Adversarial Model.** We use standard security models for secure two-party computation, which assume adversaries to be either semi-honest or malicious. Hereafter, the term *adversary* refers to protocol participants, since actions of outside adversaries can be mitigated via standard network security techniques. Following [34], protocols secure in the presence of *semi-honest adversaries* assume that parties faithfully follow all protocol specifications and do not misrepresent any information related to their inputs, e.g., size and content. However, during or after protocol execution, any party might (passively) attempt to infer additional information about other party's input. Whereas, security in the presence of *malicious parties* allows arbitrary deviations from the protocol. To ease exposition, security arguments in this paper are made with respect to *semi-honest* participants; however, efficient extensions to malicious participant security have already been developed for our cryptographic building blocks. We consider these extensions to be out of the scope of this paper.

**Security & Unlinkability.** Primitives' definitions above follow standard secure computation syntactic framework [34]. In the semi-honest model, this corresponds to considering an ideal implementation where a trusted third party (TTP) receives the inputs of both parties and outputs the result of the defined function. Protocols are secure if, in the real implementation of the protocol (without a TTP), each party does not learn more information than in the ideal implementation. Nonetheless, semi-honest security definitions might not capture the concept of *unlinkability* that refers to the impossibility for a party to learn whether any two protocol executions are related, i.e., executed by the other party on the same input.

# 3. GENODROID FRAMEWORK

As discussed in Section 1, full genome sequencing is revolutionizing diagnosis and treatment of certain diseases while producing new and more effective techniques for personalized medicine as well as ancestry and genealogy discovery. The next logical step is to transform paper-based research results into actual working computational tests available to individuals. As mentioned above, this poses challenges pertaining to usability, portability, accuracy, security, privacy and efficiency. To this end, our work focuses on the construction of efficient and privacy-preserving techniques that allow individuals to perform genomic tests while disclosing only the required minimal information to other parties. Furthermore, we aim at ubiquitous availability of genomic tests, by designing protocols that run on current (off-the-shelf) Android smartphones.[1]

## 3.1 Smartphone Rationale

We chose to focus on the smartphone environment for several reasons, chief among them is the pervasive proliferation of smartphones into many spheres of everyday life [16] and their tendency to take over tasks previously relegated to desktop or laptop computers [5]. Furthermore, the demand for smartphone use in health care applications is skyrocketing [59, 63]. Modern smartphone's unparalleled portability makes it a true anytime-anywhere computing device and its highly personal nature (even laptops are often shared) motivates using it to store private information, such as cryptographic keys, PINs/passwords as well as one's genome.

Furthermore, computational power and storage capacity of today's smartphone are comparable to those of a laptop from a few years ago. Also, smartphone vendors and mobile OS developers (e.g., Apple, Google, RIM, and Microsoft) provide programming environments that facilitate quick and efficient implementation of complex applications. From the user's perspective, smartphones are relatively easy to use and are customarily carried almost everywhere. Thus, we believe that smartphones represent a viable and an appealing platform for performing personal genomic computations.

Clearly, the smartphone is not the only choice. A genome could also be stored at a physician's office. However, an individual may visit many types of medical specialists and/or change physicians. Secure storage, replication (e.g., if a specialist needs a copy) and migration (e.g, from one physician to another) are not trivial issues. Moreover, with health care costs already very high, the public

[1] Source code for all GenoDroid applications and framework components is available at http://sprout.ics.uci.edu/projects/privacy-dna.

would be unhappy to bear the costs of additional insurance that doctors would incur in order to protect their patients' DNA. Another option is to store and process genomes on a more powerful computing device, e.g., a desktop or a laptop. In both cases, portability is a major issue since a desktop is generally stationary, whereas a laptop, though portable, is much more burdensome to carry than a smartphone. This would rule out or limit social and recreational types of genomic tests (e.g., paternity or genetic compatibility). As mentioned above, desktops and laptops can be shared by multiple users, thus, making them more vulnerable to attacks.

Alternatively, genomes could be stored in the increasingly omnipresent and (hopefully) benevolent cloud. Like a smartphone, the cloud allows anytime-anywhere access and offers computation services, in addition to storage. However, the cloud also requires reliable and pervasive Internet connectivity for its clients. A cloud vendor is also subject to unpredictable service outages and, of course, DoS/DDoS attacks. Moreover, cloud vendor privacy breaches can and should be expected; therefore, storing highly sensitive personal information in the cloud is perhaps not advisable. Even if a genome is stored in its encrypted form, unless and until fully homomorphic encryption becomes practical, complex computation on encrypted genomes stays out of reach. Naturally, in the course of a genomic test, an encrypted genome can be communicated from the cloud to the user's smartphone and the latter could perform the necessary computation. The main problem with this scenario is its cost; recall that a genome includes about $3 \cdot 10^9$ symbols. Finally, there is an issue of encryption longevity: a genome encrypted with, say, 128-bit (equivalent) key today is likely to remain secure for a few (20-25?) years. However, assuming that the cloud never "forgets" its hosted data, we need to wonder how secure would the same encryption key be 30 or 40 years from now.

## 3.2 Framework Structure

GenoDroid incorporates a number of building blocks for privacy-preserving genomic computations, e.g., decoding DNA strings produced by sequencing machines, privacy-preserving protocols, as well as auxiliary components, such as mutual authentication and device (smartphone) pairing.

One key feature of GenoDroid is its *extensibility*: although this paper focuses on only three concrete genomic tests, we are confident that our framework facilitates the development of other types of tests, without re-implementing basic components from scratch.

**Overview.** The framework currently supports two flavors of genomic tests: (i) both parties run on input of their respective entire genomes, e.g., to perform a paternity or ancestry test, or (ii) one party's input is an entire genome, while the other's – is a short sequence of *letter-position* pairs, e.g., a disease marker for personalized medicine tests. (Note that, in the latter case, letters do not have to be consecutive.)

GenoDroid integrates offline non-interactive pre-processing (e.g., on a desktop or a laptop), with online interactive computation on smartphones. Thus, computation occurs in two phases: (1) the entire genome is pre-processed, yielding a representation suitable for a given genomic test, and (2) the actual test is performed as a two-party protocol, where at least one party uses a smartphone. The pre-processing phase is particularly appealing since it separates software development from the knowledge of biological details that are not directly related to specific tests. For example, all genomic tests discussed in this paper, as well as most others, require conversion of raw output produced by a sequencing machine to a "single-string representation".[2]

### 3.2.1 Pre-processing Components

**Genomic data conversion.** Independent of the specific test, we need to convert data produced by the sequencing machine in genetic laboratory – i.e., a set of aligned strings with the associated accuracy score – to a single-string representation, where each letter in the string corresponds to the letter in the sequenced genome at the same offset. GenoDroid can support most common formats currently used by sequencing labs, i.e., SAM, BAM, FastQ, and FastA. However, in our experiments, we use BAM-formatted files downloaded from the publicly available *DNA database* [2]. This BAM format provides access to individual fragment reads, their alignment and accuracy scores, as reported by the sequencing equipment. For this particular format, our implementation uses the popular *BamTools* library [25] to access raw (binary) data.

**Test-dependent genome pre-processing.** As discussed in the rest of the paper, computational genomic tests often require an offline phase whereby the entire genome is scanned/processed to emulate *in-vitro* techniques. This is often needed to reduce the size of the input to the secure computation protocol that performs the test. GenoDroid includes a number of common DNA operations, e.g., digestion, probing, and sampling.

**Cryptographic pre-processing.** As mentioned earlier, privacy-preserving genomic tests in GenoDroid are based on two-party cryptographic protocols that yield nothing beyond intended test results. In many occasions, such protocols entail certain operations that can be pre-computed once offline. Offline costs can then be amortized over numerous (online) protocol runs. Such operations include key generation as well as protocol-specific pre-processing. In some cases, input to cryptographic pre-processing operations may depend on output of genome pre-processing (described above). Cryptographic building blocks included in GenoDroid are discussed in Section 2.3.

Each pre-processing component is assumed to be executed on a desktop or a laptop computer. (All information used in this phase must be securely erased immediately thereafter. Secure erasing is a well studied problem [36], and is out of the scope of this paper.) We utilize all available computing cores and facilitate single-pass genome processing, i.e., whenever possible, operations are executed *concurrently*, so that information is read from disk only once. We also minimize memory usage, especially during pre-processing, and do not assume that, at any time, the entire genome is stored in RAM.

### 3.2.2 Smartphone Components

Once the pre-processing phase is completed, results are transferred to an Android smartphone. All smartphone-resident GenoDroid code is written in Java and executed in the Dalvik virtual machine [35], a fast mobile-friendly JVM implementation that supports Just-in-Time compiling. As of mid-2012, high-end Android smartphones typically comes equipped with 1GB RAM and a dual- or quad-core ARM A9 processor running at 1.2-1.7GHz.

**Secure Computation.** GenoDroid implements several two-party cryptographic protocols, optimized for the Android platform: PSI-CA, SHD, and APSI. These protocol functionalities were introduced in Section 2.3, whereas, specific implemented instantiations are given in Section 6 (APSI) and in the Appendix (PSI-CA and SHD). They represent the main building blocks for the genomic

[2] Incidentally, to the best of our knowledge ours is the *first* concrete implementation of this functionality. In fact, standard genomic tools rely on a multi-string representation for human genomes, and all publicly available fully-sequenced genomes are encoded using multi-file formats (e.g., [2]).

tests presented in this paper. However, additional protocols can be easily integrated; in fact, GenoDroid already includes Private Set Intersection (PSI) from [28].

As mentioned above, we aim to minimize online (smartphone-bound), and maximize offline (desktop-bound), computation in all cryptographic protocols. However, in some cases, this can hinder protocol *unlinkability*. For example, in the SHD protocol from Section 2.3 we could minimize client's online work by pre-computing all public key encryptions. However, server would then learn whether client input changes over multiple interactions. One possible remedy is to perform pre-computation for multiple interactions. This would put a strain on the smartphone's storage. However, some operations can be pre-computed, periodically, on the smartphone too, e.g., when it is idle and connected to an external power source. On the other hand, the argument for unlinkability is, in general, not particularly convincing in the context of personalized genomic tests since one's DNA stays (almost) the same throughout one's life and, even in a social setting, one is not likely to conduct, say, a paternity test with a random stranger.

**Communication.** GenoDroid supports multiple wireless communication technologies, selectable based on specific test requirements. It currently supports secure and authenticated communication over Bluetooth, Wi-Fi and cellular networks; this includes device discovery and secure device pairing. Depending on the underlying communication technology, we use different device pairing techniques. We use Bluetooth v2.1 (or higher), which offers Secure Simple Pairing (SSP) [1] that allows two parties to bootstrap a public key authenticated channel. Over Wi-Fi, we perform both local (broadcast-based) and server-aided discovery. When using the cellular network, we also use server-aided discovery. This allows two parties, in two broadcast domains, to find a common rendezvous point. To do so, we implement a publicly available server, to which devices can advertise their presence using a human-readable ID and the fingerprint of their public key. As an alternative, the parties could identify themselves using X.509 certificates or anonymous credentials [15]: afterwards, the server reveals the counterpart's IP address and both parties can communicate directly. (If one or both parties are behind a NAT box, they would have to use the server to tunnel data.) Finally, remark that all tests presented in this paper will be instantiated over Bluetooth.

**Additional Components.** GenoDroid also includes auxiliary components that implement storage management and user interface design. Since their development prompts no research issues, we do not discuss them in detail.

# 4. PATERNITY TESTING

We use the paternity test as one of the "measuring sticks" for assessing viability of privacy-preserving genetic computations. This might not seem like a natural choice: a paternity test is not a trifle but a highly personal matter, i.e., not something we could imagine doing in a social environment. It is also usually not performed upon a routine visit to a doctor's office. Current applications of this test are generally found in legal or law enforcement settings. However, since it is arguably the most common genetic test today, we use it as a *gateway* to other types of tests. Coincidentally, as discussed below, it also happens to be the least expensive. Thus, if the targeted smartphone platform cannot handle a privacy-preserving paternity test, it would also not be able to support more complicated test types.

A genetic paternity test determines whether two individuals have a father-child relationship. In this section, we use *paternity test* to denote a protocol involving two parties (Alice and Bob) using their respective genomes as input such that they learn the binary outcome. A *privacy-preserving* version of the paternity test involves disclosing minimum amount of genomic information.

Experts claim that one individual is the father of another if the Hamming distance between their genomes is below a well-defined threshold. Thus, the two individuals could run an SHD protocol to obtain a privacy-preserving paternity test. However, this would be relatively inefficient: even without privacy, computing the Hamming distance between two whole genomes would generate traffic in the order of 1GB and require the comparison of about 3 billion elements. Such a protocol would be prohibitively expensive on smartphones. Alternatively, since about 99.5% of the human genome is the same, the two parties could, in theory, compare only the remaining 0.5%. Unfortunately, there is not enough current understanding of the genome structure to pinpoint exactly *where* this 0.5% occurs.

Our solution reduces the size of the problem by drawing from optimizations used in *in-vitro* tests. Specifically, we simulate court-admissible *in-vitro* RFLP-based paternity tests introduced in Section 2.1. DNA is digested using a set of restriction enzymes and a small number of fragments, typically between 19 and 25, is selected using probes, defined by well-known markers [30]. If two individuals are indeed father and child, then, with very high probability, the length of these fragments matches for at least a given number of fragments. This method is very reliable: with 25 fragments, its accuracy is estimated to be about 99.999% [30, 47].

There are several advantages in performing genetic paternity test computationally, rather than *in vitro*. From the privacy prospective, participants do not need to disclose to a testing lab their identity or their entire genome, nor do they have to deliver swabs to a third-party facility and wait for the outcome. Instead, they can learn the test outcome immediately. Furthermore, as recently shown in [6], RFLP paternity testing can be simulated in computation and a corresponding privacy-preserving construction can be obtained if fragment lengths are compared privately, using *Private Set Intersection Cardinality* (PSI-CA), defined in Section 2.3.

## 4.1 An Optimized Implementation

We now present GenoDroid implementation of privacy-preserving paternity test. It includes two versions: the first, similar to [6], uses PSI-CA as the underlying cryptographic building block, and the second – SHD. Our implementation supports Bluetooth as the communication channel between interacting parties, to demonstrate feasibility of our approach to location-aware, bandwidth-constrained and easy-to-bootstrap settings. However, GenoDroid seamlessly lets us choose Wi-Fi or cellular networks.

While designing this application, our main objective is to reduce online computational overhead – crucial to guarantee a positive user experience – through protocol optimization and maximal pre-processing.

**Pre-processing.** We emulate RFLP-based enzyme digestion and marker-based fragment selection, in a single pass. We design an algorithm that compares a genome to all markers in parallel. (According to genomics experts, each marker appears at most once in the entire genome, thus, as soon as a match is found, the corresponding marker is removed from the set of available ones.) As a result, a set of $n$ (in practice, 25) elements $(m_i, \ell_i)$ is produced, where $m_i$ is the $i$-th marker and $\ell_i$ is the length of the corresponding fragment. This set constitutes the input to PSI-CA/SHD protocol run on the smartphones (see below).

Furthermore, pre-processing also performs offline operations related to PSI-CA or SHD protocols. However, as discussed in Section 3.2.1, if different runs of the test must be unlinkable, we can either: (i) perform pre-computation multiple times and transfer the corresponding output to the smartphone, or (ii) let the smartphone periodically perform pre-computation when idle and connected to a power source.

**Smartphone Interaction.** Using RFLP-based techniques, we reduce privacy-preserving paternity testing to *privately comparing how many of the $n$ fragments have the same length* across the two individuals. That is, after independently applying the digestion/probing algorithm, parties learn how many fragments have the same length, and nothing else. Even if the test result is negative, this does not reveal any sensitive information. Hence, security of this construction only depends on that of the cryptographic protocol used for private comparison.

Furthermore, since input to this protocol is very small (only 25 "lengths") it can be executed efficiently. Nonetheless, we develop an optimized implementation of both PSI-CA and SHD using pipelined communication, i.e., both parties start transmitting processed data as soon as it is available, without waiting for the entire computation to finish. While one could choose any PSI-CA/SHD instantiation, our implementation (and experimental evaluation) employs the protocol from [26] (for PSI-CA) and the one based on additively homomorphic encryption (for SHD). They are both presented in the Appendix.

## 4.2 Performance Evaluation

We now evaluate GenoDroid's implementation of privacy-preserving paternity test. Let client denote the party that successfully completes Bluetooth discovery and initiates the connection, and server – the other entity. This also corresponds to the client-server nomenclature in PSI-CA and SHD protocols that we use. Recall that only client obtains the test result from the interaction, however, in our implementation, it communicates it to server over the secure channel. (This assumption does not violate our security model, since we assume that both parties are semi-honest.)

Genomes used in our experiments are downloaded from the 1,000 Genomes Project [2], and pre-processed with enzymes from [60].

Our measurements, presented in Table 1, are performed on two Nexus Galaxy phones, running Android 4.0, with a 1.2GHz TI OMAP 4460 ARM Cortex-A9 dual-core CPU, and communicating over an encrypted Bluetooth channel (as discussed in Section 3.2.2). In our experiments, we used 25 markers (corresponding to 99.999% accuracy) and tested both PSI-CA and SHD variants. Specifically, we measured:

1. ***Offline Time*** – time for both server and client to perform pre-computation pertaining to PSI-CA/SHD, on the smartphone. (Only if unlinkability is desired.)

2. ***Online Time*** – interval between the time client starts communicating with server and parties outputting the final result.

Also, we compare our optimized PSI-CA implementation with its non-optimized counterpart (without pre-processing and pipelining). Run-times are averaged over 1,000 trials to minimize measurement variations. Bandwidth measurements include all information transmitted by both client and server.

Our tests show that the optimized PSI-CA [26] yields the best results. Additively homomorphic encryption-based SHD (see the Appendix) is about 1.5 times slower than PSI-CA in the online phase. Our tests also show that pipelining and pre-computation significantly enhance user experience, since they allow the online protocol run-time to be over 3 times faster.

|  | Offline |  | Online |  |
|---|---|---|---|---|
|  | Server | Client | Time | Bandwidth |
| Optimized PSI-CA | 399 ms | 383 ms | 244 ms | 14.1 KB |
| Optimized SHD | 736 ms | 507 ms | 376 ms | 31.5 KB |
| PSI-CA no pipelining<br>no pre-computation | – | – | 784 ms | 14.1 KB |

**Table 1:** Computation & Communication costs of GenoDroid paternity test. Online cost reflect wall-clock-based run-times.

From the security point of view, the two instantiations – while both secure under the Decisional Diffie-Hellman (DDH) assumption – rely on different models: the SHD construct is secure in the standard model, whereas, selected PSI-CA is instantiated in the Random Oracle Model (ROM).

# 5. GENETIC ANCESTRY TESTING

Genetic ancestry testing allows individuals to trace their lineage through the analysis of their genomic information. Increasing understanding and availability of fully sequenced genomes makes commensurably more effective to study how susceptibility to common diseases varies among individuals and populations [61]. Also, Genome-Wide Association Studies (GWAS) [14] are gaining momentum as they study common genetic variants in different individuals to see if any variant is associated with, e.g., a disease, and possibly correlate such disease to a given ancestry line.

Besides health-related application, ancestry testing is becoming more and more popular for personal and social purposes. For instance, an increasing number of people is interested in tracing biological relatives and researching genealogical records, and discovering their family histories. Others are searching for connections to ethnic groups or geographical locations. The business side of recreational genetics is growing very fast, with scores of companies already offering ancestry tests costing only a few hundred dollars [9].

Today's genetic ancestry tests analyze either: (1) mitochondrial DNA (mtDNA), based on sequencing of maternally-inherited DNA material, or (2) the Y-chromosome, based on genomic information transmitted from father to son [9]. In both cases, individual's genomic information is compared with that of a sample individual.[3] Several commercial entities (e.g., 23andMe [3]) maintain a collection of sample genomes from individuals belonging to different ethnic groups, and compare them against their customers' genomic information to understand how they relate to known ethnic groups.

Alternatively, ancestry testing can be performed on two individuals in order to determine their genetic relationship. In this case, individuals learn whether or not they are "related" or even their distance in their common genealogical tree. Additionally, since the Y-chromosome is passed essentially unchanged from father to son, tests based on such portion of DNA provide precise information about paternal lineage. Similarly, mtDNA-based tests offer insight into one's maternal lineage.

The availability of full genome sequencing will soon allow performing more efficient computational analogs of tests that are now conducted exclusively in labs. However, privacy issues must be taken into account by both users and testing companies. Users might be unwilling to surrender their entire genomes, while companies might not wish to disclose their test details (which could represent proprietary information or trade secrets). Also, as in the case of paternity, computational genetic ancestry testing does not require parties to ship biological samples to a lab, thus, test results may be obtained significantly faster and without exposure of genetic material to third parties. Nonetheless, even without an external lab, ancestry testing between two individuals may pose privacy concerns, whenever parties may not be willing to mutually disclose

[3] For improved efficiency, rather than comparing the whole mtDNA or Y-chromosome, labs usually compare only a few (e.g., 50) SNPs across the entire genome or focus on a subset of insertions.

their ancestry or their complete genomic information. To this end, we need a privacy-preserving protocol that allows two parties to determine the *extent of their genetic proximity* without revealing any additional information about their respective genomes.

A simple way to realize a privacy-preserving ancestry test is to allow two parties to compare their entire genomes in an oblivious manner. This way, they can learn their genetic proximity without leaking additional information. However, since genomes include around three billion letters, such computation would be rather inefficient, both in computation and communication overhead. Therefore, currently popular tests restrict the comparison to either mtDNA or Y-chromosome, obtaining a slightly less accurate, yet still meaningful, metric. Nonetheless, the size of the input to the privacy-preserving computation would still be relatively large. Specifically, there are about 16,000 nucleotides in mtDNA and 58 million in the Y-chromosome. This is unlikely to yield an efficient implementation on a smartphone. A better approach combines the use of mtDNA/Y-chromosome information with either the knowledge of a subset of SNPs suitable for the test (as currently performed by commercial labs) or, if this information is unavailable, selecting a random subset as described next.

## 5.1 Our Construction

**Preliminaries.** This section presents a protocol for secure genetic ancestry testing based on Jaccard similarity index [42]. This measures the similarity of sets $A$ and $B$, as $J(A,B) = |A \cap B|/|A \cup B|$. High values of the index suggest that two sets are very similar, whereas, low values indicate that $A$ and $B$ are almost disjoint.

To realize privacy-preserving computation of $J(A,B)$, we only need secure computation of $|A \cap B|$, since $J(A,B) = |A \cap B|/(|A| + |B| - |A \cap B|)$, and this can be done using PSI-CA (see Section 2.3).

However, when two parties compute the Jaccard index, with or without privacy, they incur computation and communication complexity (at least) linear in the size of their sets. Thus, if performed over a whole genome, this computation might be relatively expensive. In fact, for any new comparison, the Jaccard index must be computed from scratch – i.e., no information used to calculate $J(A,B)$ can be re-used for $J(A,C)$. As a result, an *approximation* of the Jaccard index is often preferred, as it can be obtained at a significantly lower cost, e.g., using so-called MinHash techniques [10]. Informally, MinHash techniques extract a small representation $h_k(S)$ of a set $S$ through deterministic (salted) sampling. This representation has a constant size $\mathcal{O}(k)$, i.e., independent from $|S|$, and can be used to compute an approximation of the Jaccard index, again as the ratio between the intersection and the union of the samples. The parameter $k$ also defines the expected error with respect to the exact Jaccard index – it is bounded by $\mathcal{O}(1/\sqrt{k})$ [10].

Observe that, while the computation of $h_k(S)$ also incurs communication and computation complexity linear in set sizes, it must be performed *only once* per set, for any number of comparisons. Thus, with MinHash techniques, evaluating the similarity of any two sets requires only a constant number of comparisons. Further, we can privately approximate the Jaccard index of two sets by executing PSI-CA on input MinHash samples (and not the entire sets).

**Ancestry Testing.** Using MinHash, we implement an efficient privacy-preserving genetic ancestry testing protocol that can be executed on smartphones. Our protocol leverages a pre-processing phase performed on a desktop or laptop computer. The pre-processing phase in our construction takes as input the set representation of an individual's mtDNA or Y-chromosome. It extracts a compact representation, using MinHash, and also performs the offline computation phase for the PSI-CA protocol of [26]. Finally, the two parties perform the (PSI-CA) online computation on their smartphones and obtain the test result, i.e., estimate *how similar* their genomes are, based on one of the following datasets: (1) a small selection of SNPs; (2) an entire Y-chromosome; (3) the whole mtDNA material; (4) all known SNPs (approximately 3 million).

## 5.2 Implementation Details

Our implementation realizes privacy-preserving genetic ancestry testing by privately (and probabilistically) comparing how many common SNPs, mtDNA or Y-chromosome base pairs two individuals share, using MinHash and PSI-CA from [26] (in the Appendix).

**Pre-processing.** Depending on whether the test is performed using SNPs, mtDNA or Y-chromosome, the offline phase of our protocol – performed on a desktop computer – involves slightly different computation. SNP-based testing requires iterating over the entire genome (about three billion base pairs) to extract all known SNPs. The output of this phase is composed of around 3 million elements, which we represent as $(b_i||loc_i)$ where $b_i$ corresponds to the $i$-th SNP and $loc_i$ to its position in the genome. The set $\{(b_1||loc_1), \ldots, (b_n||loc_n)\}$ is then used as input for the offline phase of the PSI-CA protocol in [26].

The pre-processing phase of both mtDNA- and Y-chromosome based tests consists in the offline phase of selected PSI-CA instantiation, executed on input the whole mtDNA (16,000 base pairs) or the Y-chromosome (58 million nucleotides), represented as in the SNP-based test.

**Smartphone Interaction.** Regardless of the dataset used for the test, the online computation involves the comparison of a subset of the parties' input, extracted using MinHash. Once they establish a connection, two parties negotiate a common salt, which is used by MinHash to extract $k$ elements from their respective input. These $k$ elements are then used as the input to the PSI-CA protocol.

## 5.3 Performance Evaluation

We evaluate GenoDroid's implementation of privacy-preserving ancestry test using the same setup as in Section 4.2. We measure the offline overhead as the time required to perform the PSI-CA precomputation. Online cost is measured as the online part of PSI-CA. We also include time, bandwidth and pre-computation storage requirements of the protocol executed without the use of MinHash on the entire genome, mtDNA, Y-chromosome and all SNPs. We also perform our measurements on input 50 randomly-selected SNPs, to simulate the test performed by 23andMe (note that the actual SNPs used by 23andMe are not publicly known).

Measurements are presented in Table 2. We instantiate MinHash with $k = 10{,}000$, leading to an error of about 1% in the final result. Run times are averaged over 1,000 trials to minimize measurement variations.[4] Pre-computation storage requirements for MinHash are identical to that of performing the same test without MinHash – i.e., 8.5 MB for mtDNA, 366 MB for SNPs and 6.9 GB for Y-chromosome tests. Bandwidth measurements include all information transmitted by both parties. Similar to paternity test, client denotes the party that successfully completes Bluetooth discovery and initiates the connection, and server – the other entity, and this also corresponds to client and server nomenclature in PSI-CA [26].

Our experiments show that, without using MinHash, only mtDNA-based tests can be executed on smartphones in a reasonable amount of time and with acceptable communication overhead. Due to our choice of $k$, the use of MinHash improves the performance of all

[4] For tests longer than one day, running times have only been estimated by running tests on smaller inputs – this was possible as tested protocols incur linear complexities.

|  | Offline |  |  | Online |  |
|---|---|---|---|---|---|
|  | Pre-comp. Size | Server | Client | Time | Bandwidth |
| Full Genome | 358 GB | 494 days | 494 days | 273 days | 1544 GB |
| Y-chromosome | 6.9 GB | 9.5 days | 9.5 days | 5.3 days | 29.9 GB |
| All SNPs | 366 MB | 11.9 hours | 11.9 hours | 6.6 hours | 1.5 GB |
| mtDNA | 8.5 MB | 227.7 s | 227.0 s | 125.3 s | 8.5 MB |
| MinHash (all tests) | – | – | – | 220.6 s | 5.2 MB |
| MinHash w/ pre-comp. (all tests) | depends on test | 142.3 s | 141.9 s | 78.3 s | 5.2 MB |
| 50 SNPs (23andMe) | 6.25KB | 713 ms | 711 ms | 394 ms | 27 KB |

**Table 2:** Computation & Communication Costs of our Privacy-Preserving Ancestry Test.

protocols. Specifically, it takes, 78 seconds with (and 220 seconds without) pre-computation, to execute privacy-preserving genetic ancestry testing between two parties equipped with Android smartphones, with *all* datasets. (In fact, $k$ is a constant, thus, it is independent of size of the original sampled set).

The amount of space required to store precomputed values prompts an interesting tradeoff. While it is easy to justify the use of 8.5 MB of memory for storing the elements precomputed from mtDNA genetic material, users may prefer not to store 366 MB or 6.9 GB of data (in the case of SNPs and Y-chromosome respectively) and rather perform the whole PSI-CA protocol online, incurring an additional two minutes of computation.

It is interesting to observe that our simulation of the 23andMe tests shows that, with the appropriate knowledge, genetic ancestry testing can be performed in near real time on current smartphones.

## 6. PERSONALIZED MEDICINE

Recall that PM aims at identifying genomic information needed to accurately predict: (1) a susceptibility to a given disease, (2) the course of a disease, and (3) response to treatment. For example, before treating leukemia and other autoimmune diseases, physicians are required, by the US Food and Drug Administration (FDA), to test patients for certain mutations of the *tpmt* gene. This particular gene codes the enzyme responsible for metabolization of a number of drugs. Thus, sensitivity and toxicity response to these drugs varies according to *tmpt* mutations. In general, PM entails testing for numerous genetic markers, ranging from one to a few hundred mutations.

With growing availability and better understanding of genomic information, future health-care will be tailored to the patient's DNA and genetic PM tests are soon likely to become commonplace. This prompts some concerns about the entities (e.g., hospitals, doctors, insurance carriers and labs) that could obtain genomic information to perform such tests. Besides obvious privacy issues due to bulk access to patients' genomes, there is a more subtle problem involving liability and long-term safety of genomic data. We alluded to it earlier, in Section 3.1. Entities handling genomic data need to demonstrate that it was treated appropriately and disposed of when no longer needed. Storing genomic information, even for a short time, creates potentially attractive targets for breaches and attacks.

One intuitive approach is to let the patient independently run specialized software over her genome and check for a match (or lack thereof) against a given drug's fingerprint. However, pharmaceuticals usually consider DNA fingerprints of their drugs to be trade secrets and thus are not expected to reveal them. At the same time, for every new drug, pharmaceuticals are required to obtain approval from some government agency, e.g., the Food and Drug Administration (FDA) in the United States. Therefore, an ideal scenario would be a privacy-preserving PM test, whereby: (1) the patient learns the outcome but not the drug's DNA markers, (2) the patient is somehow assured that the drug has been authorized by the relevant government agency, and (3) neither the pharmaceutical nor any other party learns any information about the patient's genome.

The rest of this section presents a technique for privacy-preserving PM testing, implemented in GenoDroid. Similar to other GenoDroid tests, it runs in real-time and involves a patient with an Android smartphone and a tester, e.g., a medical lab or a physician's office. We begin by discussing a strawman construction that, albeit privacy-preserving, is not suitable for a smartphone platform and proceed to illustrate our practical cloud-aided approach.

### 6.1 Initial Construction

Similar to [6], our initial approach to privacy-preserving PM testing relies on APSI, defined in Section 2.3. We use the particular construction from [27] where the patient plays the role of server, the tester – client, and the FDA – mutually trusted CA. The APSI protocol from [27] is secure in the malicious model. However, we argue that malicious player security may be not needed for the PM setting that usually takes place in a medical lab or a physician's office. This is because there is generally some basic trust between a patient and a doctor (or a lab performing the test). Also, computational tests can be audited, e.g., if protocol transcripts are mandated to be kept. Furthermore, there could be severe consequences for malicious behavior, e.g., loss of medical license. Thus, we slightly modify the construction from [27] to only achieve semi-honest security.

Specifically, protocol executes on common input CA's RSA public key $(N, e)$, a generator $g$ of $\mathbb{QR}_N$, and two cryptographic hash functions $H, H'$, modeled as random oracles. (All computation is performed $\bmod N$). For each element $c_i \in \mathcal{C} = \{c_1, \ldots, c_v\}$ in its input, client obtains $\sigma_i$ such that ${\sigma_i}^e = H(c_i)$, i.e., a CA-issued signature warranting authorization. The interaction starts with client sending $\{a_i = \sigma_i \cdot g^{R_{c:i}}\}_{i=1}^{v}$ (for $R_{c:i} \leftarrow \mathbb{Z}_{N/2}$) to server. Then, server selects $R_s \leftarrow \mathbb{Z}_{N/2}$ and returns $(Z = g^{2eR_s}, \{a'_i = {a_i}^{2eR_s}\}_{i=1}^{v})$. Also, for each element $s_j \in \mathcal{S} = \{s_1, \ldots, s_w\}$ in its input, server sends $\{ts_j = H'(H(s_j)^{2R_s})\}_{j=1}^{w}$ to client. Finally, client computes $\{tc_i = H'(a'_i \cdot Z^{-R_{c:i}})\}_{i=1}^{v}$ and outputs intersection $\mathcal{S} \cap \mathcal{C} = \{c_i \in \mathcal{C} \mid tc_i \in \{ts_j\}_{j=1}^{w}\}$.

APSI-based privacy-preserving PM testing is as follows. The patient uses her smartphone that stores pre-processed genomic information as well as various cryptographic material. We assume that, well ahead of running the protocol and after positive clinical trials, the FDA granted authorization ($auth$) to the pharmaceutical for a specific DNA fingerprint (*fp*), corresponding to a genomic test. We denote $fp = \{(b^*_j || j)\}$, where each symbol $b^*_j$ is expected to occur at position $j$ of a fully sequenced genome, and $auth = \{H(b^*_j || j)^d \bmod N\}$, where $N$ and $d$ are FDA's RSA modulus and private key, respectively. The patient's private input is her entire genome. (This is unavoidable since the test fingerprint can occur anywhere.) The tester's input is $(fp, auth)$, as defined above.

Tester and patient engage in APSI protocol, as described above, and, at the end of the protocol, the tester learns whether the patient's genome matches *fp*, provided that $auth$ is a valid authorization of *fp*. Furthermore, the tester learns nothing further about the patient's genome, and (2) the patient learns nothing about *fp* or $auth$.

We note that server-side (patient's) computation can be partitioned into *offline* and *online* phases. Specifically, $\{ts_j\}_{j=1}^{w}$ can be pre-computed once for any number of tests. Therefore, while offline computation is linear in the size of the genome, its online counterpart is linear in the number of tested *loci*. However, $\{ts_j\}_{j=1}^{w}$ values must still be transferred, implying that online communication complexity is linear in the genome size. This translates into several gigabytes and obviously hinders adoption on modern smartphones.

## 6.2 Cloud-Aided Variant

We now show how to modify the initial approach to make it amenable for smartphones. The main idea is to off-load some of the patient's (server's) computation to the cloud. However, we immediately acknowledge that involving the cloud triggers the risks discussed at the end of Section 3.1.

The patient needs to pre-process the genome and upload the result to a (semi-honest) cloud provider. Clearly, such pre-processing must include some form of encryption, to conceal the genome from the cloud. Moreover, the encrypted elements must be shuffled in order to (partially) hide access patterns from the cloud. When the test is conducted, the patient explicitly grants the tester (client) access to cloud-resident genomic information, such that only the test result is learned. This operation must be efficient and should prevent the tester from colluding with the cloud provider and learning any additional genomic information.

Unlike paternity and ancestry testing, the pre-processing phase in this cloud-aided variant results in two output sets: one uploaded to the patient's smartphone, and the other – to the cloud provider. Despite the presence of the cloud provider, the actual protocol is still based on APSI from [27].

**Pre-processing.** Besides genomic pre-processing (e.g., format transformation), patient pre-computes, e.g., on a desktop, $\{ts_j = H'(H(b_j^*||j)^{2R_s})\}_{j=1}^{w}$ and uploads the result, after shuffling, to the cloud. This does not reveal any information about the genome. Then, public cryptographic parameters $(N, e, g, H, H')$, along with private random value $(R_s)$, are uploaded to the patient's smartphone. (Note that there is no pre-computation on the tester side.)

**Interaction.** This phase transpires between a smartphone-equipped patient and a tester on a desktop or a laptop connected to the Internet. Together, they execute the online phase of APSI, i.e., the tester sends the patient $\{a_i\}_{i=1}^{v}$ and receives $\{a'_i\}_{i=1}^{v}$. The tester then computes $\{tc_i\}_{i=1}^{v}$ and finds matching $ts_j$-s using the cloud provider. There are at least three ways for tester to do so:

1. Query the cloud using $\{tc_i\}_{i=1}^{v}$.
2. Download all $\{ts_j\}_{j=1}^{w}$ from the cloud.
3. Privately query the cloud using, e.g., Private Information Retrieval (PIR) [22].

GenoDroid currently implements (1) since it is the most efficient of the three. Of course, since the tester is running on a regular computer, we could choose option (2). However, even on a desktop with a wired Internet connection, downloading approximately 60GB of data is time-consuming for the tester and unscalable for the cloud provider.[5] As far as option (3), recent advances in, and optimization of, single-server PIR techniques have yielded promising results that might be efficient enough for the tester's desktop or laptop, especially if queried database is maintained in a cloud cluster [8]. However, since our emphasis in this paper is on the smartphone platform, we leave this item for future work.

**Threat Model and Security.** We assume that each participant is semi-honest, i.e., interacts with the other two participants while following all protocol specifications. However, a participant might attempt to surreptitiously learn further information about others' inputs. The underlying APSI protocols, as mentioned earlier is secure in the semi-honest model, with two parties. In the presence of the cloud provider (the 3rd party), collusions should be considered. If a tester colludes with a cloud provider, the only danger is that the former might obtain the entire "encrypted" genome – i.e., does not learn any meaningful information. Whereas, if a patient colludes with a cloud provider, they could learn some information about the proprietary (to the pharmaceutical) DNA fingerprint *fp*. In particular, the colluding parties could learn which (letters, positions) of the patient's DNA are specified in *fp*. However, this can occur only for the portion of the fingerprint matching the patient's genome. The case of a tester colluding with a patient might seem uninteresting; however, even though a tester acts on behalf of a pharmaceutical, their collusion can result in leakage of portions of *fp*, similar to the previous case.

| | Online | | Bandwidth | |
|---|---|---|---|---|
| | Patient | Tester | Tester-Cloud | Tester-Patient |
| *hla-b*5701* | 78 ms | 141 ms | 0.40 KB | 0.40 KB |
| *tpmt* | 187 ms | 301 ms | 1.77 KB | 1.77 KB |

**Table 3:** Computation & Communication costs of GenoDroid's cloud aided hla-b and tpmt tests. Online cost reports wall-clock running time.

In summary, we envision multiple entities (including generic cloud providers, HMO-s and insurance companies) will offer genomic cloud services to the public. Our foray into privacy-preserving personalized medicine testing is just one example of what can be achieved by combining the power of cloud computing/storage with smartphones in genomic computation and calls for further research.

## 6.3 Performance Evaluation

We now evaluate GenoDroid's implementation of privacy preserving PM testing. To compare with previous work, we use the same genetic fingerprints as in [6], i.e., the ones describing mutations *hla-B*5701* and *tpmt*. The former is associated with extreme sensitivity to abacavir, an HIV drug, and its fingerprint is composed of 2 nucleotide positions; the latter is tested before prescribing 6-mercaptopurine to leukemia patients. The *tpmt* fingerprint contains 6 nucleotide positions.

Measurements presented in Table 3 were obtained using the same setup as in Section 4.2, i.e., we used two Android smartphones communicating over Bluetooth. As discussed in Section 6.2, the tester can use a desktop computer and parties could communicate over Wi-Fi. However, we use Bluetooth for consistency's sake and in order to provide conservative results: our measurements represent an upper bound for the cost in a real-world setting.

Patient and tester interact via an APSI protocol where the patient acts as server, and the tester – client. Client does not perform any precomputation, and we assume that server uploads its encrypted genome to the cloud ahead of time. Measurements reported in Table 3 reflect the online phase of the APSI protocol. The Cloud does not perform any cryptographic operation, and therefore, we do not consider any of its computation costs. Bandwidth is measured for tester's interaction with patient and with cloud separately (bandwidth measurements are independent of the transmission medium).

Our experiments show that both tests for *tpmt* and *hla-b* can be executed on smartphones in well under a second. Our implementation compares favorably with that of [6] in terms of both bandwidth – due to the use of cloud-aided computation – and offline computation. Although our online phase runs slower than experiments reported in [6], we emphasize that (1) our timing include not only the cryptographic blocks, but rather a whole implementation over networked devices; (2) the Android smartphones used in our tests are significantly slower than the desktop platform used in [6].

# 7. USABILITY STUDY

One of the goals of the GenoDroid framework is to guarantee portability and accessibility, to average non-tech-savvy users, of

[5] GenoDroid assumes 20-byte $H'()$, thus, since the genome contains approx. $w = 3 \cdot 10^9$ nucleotides, $\{ts_j\}_{j=1}^{w}$ values amount to $(20 \cdot 3 \cdot 10^9)$B$=$60GB.

privacy-preserving genomic tests. Thus, usability is one of the key factors influencing its acceptance. To this end, we conducted a usability study to obtain feedback on our prototype applications as well as to assess the sensitivity of subjects to privacy concerns related to their genomic information.[6]

The study was conducted on 16 subjects (8 male and 8 female), sampling a diversified population of students, researchers, and non-scientific personnel. 90% of subjects belong to the 25-to-40 age range. Applications were run over mock human genomes, with same length and comparable nucleotide distribution as real human genomes. Our test mock-up was implemented using two Android-equipped Samsung Galaxy Nexus phones and used Bluetooth as the wireless communication medium.

As discussed in Section 4, we use *paternity test* as one of the "measuring sticks" for assessing viability and usability of privacy-preserving genetic computations. Arguably, paternity test is the most common genetic test today and non-tech-savvy users would likely relate to it more than to any other test. (The actual app will be released with the final version of the paper.) The chain of events is as follows: since the pre-computing phase takes place on desktops, the app is pre-loaded with the result of the RFLP-driven digestion and probing. Once the app is launched, the user is prompted with the Android interface to carry out secure Bluetooth pairing with another device and establish an authenticated and encrypted channel. Upon successful connection establishment, both users see a *"Start Test"* button; the first user to click is prompted with a *"Waiting for other party"* message, and, after the counterpart also hits start, the test is initiated. Finally, users are displayed with the test result, i.e., *"Tested individuals are/are not father and child"*. (In our user studies, the result is negative.)

After the test, subjects were asked to fill out a questionnaire corresponding to Brooke's well-known 10-item System Usability Scale (SUS) [11], where answers indicate the degree of agreement or disagreement with the corresponding statement on a 5-point scale. Subjects rated application's usability, on average, at 82/100 on the SUS scale. Such a high score can be safely considered above industry average and confirms that the perceived usability of the software is high enough to be deployed in practice. This is not surprising since the application is straightforward to use, does not require any high-end technical skill, and test running time is extremely low (in fact, a delay of 250ms is barely noticeable by users, and provides a seamless experience [57]).

We also asked subjects to answer a few questions about privacy concerns related to genomic information. Subjects were asked to indicate their agreement on a statement using a 5-point scale, where 1 corresponds to "strongly disagree" and 5 – to "strongly agree". Due to space limitation, we defer complete discussion of this part to the full version of the paper. However, we report a few interesting findings. Subjects were "concerned with potential privacy exposure of (their) genomic information" (average 4.21/5 agreement). Somewhat more surprising is that our test subjects were "concerned with privacy even if tests are beneficial to (their) health" (average 3.08 agreement), while they were "in favor of genetic tests if they do not invade (their) privacy" (4.81 average agreement). Note that subjects in our study are *not* privacy/security researchers.

## 8. RELATED WORK

**Secure Testing on Fully Sequenced Human Genomes.** Non-cryptographic approaches to privacy, such as de-identification, are often ineffective on genomic data, as shown by some recent studies [49, 38, 72, 76]. As a result, several privacy-preserving cryptographic techniques have been proposed. We now review techniques for secure testing on fully sequenced human genomes.

Baldi, et al. [6] recently introduced several cryptographic protocols for privacy-preserving testing of fully sequenced human genomes, including RFLP-based paternity test and genetic screening for personalized medicine or recessive genetic diseases. Similar to our setting, individuals obtain their genomes and allow authorized parties (e.g., doctors) to run genetic tests such that only test results are disclosed to one or both parties. However, [6] only addresses the issue of designing cryptographic protocols, and does not deal with real-world issues. In particular: genome conversion, extensibility, fine-grained optimizations on mobile devices and usability, are not considered. Whereas, our work provides a set of working practical instantiations of genomic tests on a popular smartphone platform. Also, in contrast with the PM technique in [6], our work includes a cloud-aided variant that facilitates much more efficient operation. Finally, we design and implement new operations, such as privacy-preserving genetic ancestry testing.

Chen, et al. [21] studied the problem of privacy-preserving mapping and aligning of human genomic sequences to a reference genome, by outsourcing work to the cloud and protecting sensitive DNA information. Since [21] does not consider genomic testing, it is orthogonal to our work.

**Secure Computation on DNA Fragments.** We now review prior work realizing secure computation on DNA fragments, as opposed to fully sequenced genomes.

Bruekers, et al. [13] presented privacy-preserving techniques for some DNA operations, based on Short Tandem Repeat (STR). Proposed techniques use homomorphic encryption on DNA fragments to perform comparisons. Testing protocols are resilient to small numbers of errors, however, their complexity increases with the number of tolerated errors [7]. Also, [13] leaves as an open problem the scenario where an attacker faithfully runs the protocol but with arbitrary inputs. In this setting, an attacker, given STR's limited entropy, can "lie" about its STR profiles and run multiple dependent protocols, thus reconstructing the other party's profile.

Wang, et al. [73] developed techniques for computation on genomic data stored at a data provider, including: edit distance, Smith-Waterman and search for homologous genes. Program specialization is used to partition genomic data into "public" (most of the genome) and "sensitive" (a very small subset of the genome). Sensitive regions are replaced with symbols by data providers (DPs) before data consumers (DCs) have access to genomic information.

Troncoso-Pastoriza, et al. [71] proposed an error-resilient privacy-preserving protocol for string searches. One party, on input of its DNA snippet, can verify the existence of a short template (e.g., a genetic test held by the service provider) within its (short) snippet. This technique handles errors and maintains privacy of both the template and the snippet. Each query is represented as an automaton executed using a finite state machine (FSM) in an oblivious manner. However, the number of FSM states is always revealed to all parties.

The work of Katz, et al. [46] also considers DNA testing. Specifically, it realizes secure computation of the CODIS test [70] (run by the FBI for DNA identity testing), that could not be otherwise implemented using pattern matching or FSM.

Another set of cryptographic results focus on privately computing the *edit distance* for two strings $\alpha, \beta$. (Edit distance is defined as the minimum number of operations, such as, delete, insert, or replace, needed to transform $\alpha$ into $\beta$.) Privacy-preserving computa-

[6] Our study received the "Exempt Registration" status from UC Irvine's Institutional Review Board (IRB).

tion of Smith-Waterman scores [67] has also been investigated and used for sequence alignment. Jha, et al. [43] show how to securely compute edit distance using garbled circuits [75], and demonstrate that the resulting overhead is acceptable only for small strings (e.g., a 200-character strings require 2GB circuits).

**Secure Computation on Mobile Devices.** In [39], Huang, et al. present a preliminary analysis of the performance of pipelined garbled circuits on smartphones and deploy optimized circuit-based techniques for secure computation [40]. [39] mentions *personal genetics* as a possible application, showing that two individuals could compare 25 genetic features (e.g., common recessive genetic diseases) in about 7 seconds. Although, in this paper, we do not focus on this test, we observe that only if *all* known recessive diseases are compared, the test becomes truly privacy-preserving. (In fact, an individual who requests a specific subset of tests may be revealing the disease he/she suffers). While it may be reasonable today to assume that the number of known recessive diseases is in the order of 25, this may soon become unrealistic, as full genome sequencing continuously enables better understanding of the genome and discovery of new diseases. Also, [39] presents, as an application example, *CommonContacts*, an Android app that privately discovers the contacts common between two users. It realizes Private Set Intersection [32], where sets correspond to contact lists. However, computation overhead appears to be still too high, in practice, to scale up to fully sequenced genomes, since executing *CommonContacts* on lists with 256 entries takes about 10 minutes [39].

Carter, et. al [19] presented the concept of *Efficient Mobile Oblivious Computation* (EMOC), i.e., a technique that completely replaces garbled circuits with homomorphic operations on ciphertexts. EMOC is used to solve Yao's millionaires problem [75] and compute common friends in a social network. Finally, Mood, et al. [53] have recently proposed a memory optimization for garbled-circuit based applications for generic secure computation on mobile phones.

# 9. CONCLUSIONS AND FUTURE WORK

This paper explored the viability and practicality of privacy-agile computational genomic tests in the portable and pervasive setting of modern smartphones. We combined domain knowledge in biology, genomics, ubiquitous computing, and applied cryptography, to design and build a personal genomic toolkit, called GenoDroid. We implemented it on the Android platform, assessed its performance and conducted pilot usability study that produced some encouraging results. We certainly plan to incorporate support for additional genetic tests in GenoDroid, e.g., privacy-preserving organ donor-recipients genetic compatibility testing. We also intend to look into computational genetic tests for non-human digitized genomes, e.g., plants as well as pets and livestock. Last but not least, lots of work remains to be done in terms of user perception (and general usability) of personal computational genetic tests.

**Acknowledgments.** We are grateful to Pierre Baldi, Roberta Baronio, Gregory Norcie, and Elaine Shi for their valuable feedback and useful hints.

# 10. REFERENCES

[1] Bluetooth SIG, Simple Pairing Whitepaper. http://preview.tinyurl.com/bluetooth-simple-pairing, 2007.

[2] 1000 Genomes Project. A Deep Catalog of Human Genetic Variation. http://www.1000genomes.org/.

[3] 23andMe. https://www.23andme.com/.

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

[5] AT&T. Mobilizing Enterprise Applications. http://www.business.att.com/content/whitepaper/mobilizing-enterprise-applications.pdf, 2010.

[6] 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.

[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] D. Bolnick et al. GENETICS: The Science and Business of Genetic Ancestry Testing. *Science*, 318(5849), 2007.

[10] A. Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences*, 1997.

[11] J. Brooke. SUS-a quick and dirty usability scale. *Usability evaluation in Industry*, 189, 1996.

[12] T. Brown. *Genomes 3*. Garland science, 2006.

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

[14] P. Burton et al. Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls. *Nature*, 447, 2007.

[15] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. *Eurocrypt*, 2001.

[16] Canalys Research. Smart phones overtake client PCs in 2011. http://www.canalys.com/newsroom/smart-phones-overtake-client-pcs-2011.

[17] T. Canli. The emergence of genomic psychology. *Nature*, 8, 2007.

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

[19] H. Carter, C. Amrutkar, I. Dacosta, and P. Traynor. Efficient Oblivious Computation Techniques for Privacy-Preserving Mobile Applications. Technical report, 2011. http://smartech.gatech.edu/handle/1853/42367.

[20] S. Cass. Cheap DNA sequencing will drive a revolution in health care. http://www.technologyreview.com/biomedicine/24587/, 2010.

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

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

[23] F. Collins. Medical and societal consequences of the human genome project. *New England Journal of Medicine*, 341(1), 1999.

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

[25] D. Barnett. BamTools. https://github.com/pezmaster31/bamtools.

[26] E. De Cristofaro, P. Gasti, and G. Tsudik. Fast and Private Computation of Set Intersection. Cryptology ePrint Archive, 2011. http://eprint.iacr.org/2011/141.

[27] E. De Cristofaro, J. Kim, and G. Tsudik. Linear-complexity private set intersection protocols secure in malicious model. In *Asiacrypt*, 2010.

[28] E. De Cristofaro and G. Tsudik. Practical Private Set Intersection Protocols with Linear Complexity. In *FC*, 2010.

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

[30] D. Endean. RFLP analysis for paternity testing: observations and caveats. In *Human Identification*, 1989.

[31] J. Fowler, J. Settle, and N. Christakis. Correlated genotypes in friendship networks. *National Academy of Sciences*, 108(5), 2011.

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

[33] M. Fumagalli et al. Parasites represent a major selective force for interleukin genes and shape the genetic predisposition to autoimmune conditions. *Experimental Medicine*, 206(6), 2009.

[34] O. Goldreich. *Foundations of cryptography: Basic applications*, chapter 7.2.2. Cambridge Univ Press, 2004.

[35] Google. Dalvik. http://code.google.com/p/dalvik/.

[36] P. Gutmann. Secure Deletion of Data from Magnetic and Solid-state Memory. In *Usenix Security*, 1996.

[37] M. Hoffman. The genome-enabled electronic medical record. *Journal of Biomedical Informatics*, 40(1), 2007.

[38] 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.

[39] Y. Huang, P. Chapman, and D. Evans. Privacy-preserving applications on smartphones. In *HotSec*, 2011.

[40] Y. Huang, D. Evans, J. Katz, and L. Malka. Faster secure two-party computation using garbled circuits. In *Usenix Security*, 2011.

[41] International Human Genome Sequencing Consortium. Initial sequencing and analysis of the human genome. *Nature*, 409, 2001.

[42] P. Jaccard. Etude comparative de la distribution florale dans une portion des Alpes et du Jura, 1901.

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

[44] J. Kaiser. A plan to capture human diversity in 1000 genomes. *Science*, 319, 2008.

[45] J. Katz and Y. Lindell. *Introduction to modern cryptography*. Chapman & Hall/CRC, 2008.

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

[47] E. Lander. DNA fingerprinting on trial. *Nature*, 339(6225), 1989.

[48] R. Lewis and A. Reynolds. *Human genetics: concepts and applications*. McGraw-Hill, 2003.

[49] 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.

[50] B. Malin and L. Sweeney. Determining the identifiability of DNA database entries. In *AMIA*, 2000.

[51] B. Malin and L. Sweeney. Re-identification of DNA through an automated linkage process. In *AMIA*, 2001.

[52] A. Menezes, P. V. Oorschot, and S. Vanstone. *Handbook of applied cryptography*. CRC, 1997.

[53] B. Mood, L. Letaw, and K. Butler. Memory-Efficient Garbled Circuit Generation for Mobile Devices. In *FC*, 2012. http://fc12.ifca.ai/pre-proceedings/paper_71.pdf.

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

[55] National Center for Biotechnology Information (US). Restriction Fragment Length Polymorphism (RFLP). http://www.ncbi.nlm.nih.gov/projects/genome/probe/doc/TechRFLP.shtml, 2011.

[56] NHGRI. DNA Sequencing Costs – Data from the NHGRI Large-Scale Genome Sequencing Program. http://www.genome.gov/sequencingcosts, 2012.

[57] J. Nielsen. *Usability Engineering*. 1997.

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

[59] G. Putzer and Y. Park. Are Physicians Likely to Adopt Emerging Mobile Technologies? Attitudes and Innovation Factors Affecting Smartphone Use in the Southeastern United States. *HIM*, 2012.

[60] R. Roberts. REBASE, The Restriction Enzyme Database. ftp://ftp.neb.com/pub/rebase/commdata.txt.

[61] C. Rotimi and B. Jorde. Ancestry and disease in the age of genomic medicine. *The New England journal of medicine*, 363(16), Oct. 2010.

[62] F. Sanger et al. The nucleotide sequence of bacteriophage $\varphi$X174. *Journal of molecular biology*, 125(2), 1978.

[63] J. Sarasohn-Kahn. *How smartphones are changing health care for consumers and providers*. California HealthCare Foundation, 2010.

[64] E. Schadt, S. Woo, and K. Hao. Bayesian method to predict individual SNP genotypes from gene expression data. *Nature Genetics*, 2012.

[65] E. Singer. Democratizing DNA Sequencing. http://www.technologyreview.com/biomedicine/26850, 2012.

[66] N. Siva. 1000 Genomes project. *Nature biotechnology*, 26(3), 2008.

[67] T. Smith and M. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147, 1981.

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

[69] H. Tabor, B. Berkman, S. Hull, and M. Bamshad. Genomics really gets personal: How exome and whole genome sequencing challenge the ethical framework of human genetics research. *American Journal of Medical Genetics*, 2011.

[70] The Federal Bureau of Investigation. Combined DNA Index System (CODIS). http://www.fbi.gov/about-us/lab/codis.

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

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

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

[74] A. Weston and L. Hood. Systems biology, proteomics, and the future of health care: toward predictive, preventative, and personalized medicine. *Journal of proteome research*, 3(2), 2004.

[75] A. Yao. Protocols for secure computations. In *FOCS*, 1982.

[76] 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: Protocol Instantiations

**Private Set Intersection Cardinality (PSI-CA).** We implement the PSI-CA protocol of [26], secure in the semi-honest model under the DDH assumption in Random Oracle Model (ROM). It executes on common input two large primes $p, q$, s.t. $q|p-1$, and two cryptographic hash functions $H$ and $H'$. We assume that $p$, $q$, $H$ and $H'$ are publicly available and that the same values are used in all instantiations of a specific genomic application in order to enable pre-computation. (Computation below is assumed $\bmod\ p$.)

Client picks $R_c$ randomly from $\mathbb{Z}_q$ and then, for each $c_i \in \mathcal{C} = \{c_1, \ldots, c_v\}$, computes $a_i = H(c_i)^{R_c}$. Then it sends $\{a_i\}_{i=1}^{v}$ to server. The latter picks $R_s$ and randomly from $\mathbb{Z}_q$ and sends client $\{a_i' = (a_i)^{R_s}\}_{i=1}^{v}$ after being shuffled. Server also sends, for each $s_j \in \mathcal{S} = \{s_1, \ldots, s_w\}$, $ts_j = H'(H(s_j)^{R_s})$.

Finally, client outputs $|\mathcal{S} \cap \mathcal{C}|$ as:

$$|\{ts_1, \ldots, ts_w\} \cap \{H'({a_1'}^{1/R_c}), \ldots, H'({a_v'}^{1/R_c})\}|.$$

**Secure Hamming distance (SHD).** To obtain the SHD of two equal-length string in the semi-honest model, we can use any additively homomorphic encryption scheme, such as, Paillier [58] or additive ElGamal variant [29].

Client generates keypair $(pk, sk)$ and, given string $\mathsf{C} = c_1 || \ldots || c_n$, computes $\{a_i = E_{pk}(-c_i)\}_{i=1}^{n}$. Similarly, server computes $\{b_i = E_{pk}(s_i)\}_{i=1}^{n}$. Client sends $\{a_i\}_{i=1}^{n}$ to server, which computes $d_i = b_i * a_i$, such that $d_i$ is the encryption of $(s_i - c_i)$. Next, server picks $n$ random values, $\{r_i\}_{i=1}^{n}$, and returns $\{e_i = {d_i}^{r_i}\}_{i=1}^{n}$ to client, after shuffling them. Finally, client sets $z_i = 1$ if $D_{sk}(e_i) = 0$, and $z_i \neq 0$ otherwise, and computes $\mathsf{HD}(\mathsf{S}, \mathsf{C}) = \sum_{i=1}^{n}(z_i)$.

Our implementation uses the additive ElGamal encryption scheme, secure under the Decisional Diffie-Hellman (DDH) assumption in the standard model. Let $p$, $q$ (s.t., $q|p-1$), and a generator $g$ of a subgroup of $\mathbb{Z}_p^*$ of order $q$, be public parameters. Let $x \leftarrow \mathbb{Z}_q$ be the private key $sk$ and $y = g^x \mod p$ the public key $pk$. ($pk$ is transferred to server as the first step of the interaction). Encryption of message $m \in \mathbb{Z}_q$ is $Enc_{pk}(m) = \langle c_1, c_2 \rangle = \langle g^r, h^r \cdot g^m \rangle$ for $r \leftarrow \mathbb{Z}_q$. Decryption is $Dec_{sk}(Enc_{pk}(m)) = c_2 / {c_1}^x = g^m$, thus, one can efficiently test whether $Enc_{pk}(m)$ is an encryption of 0.