Peter Ney, Luis Ceze, Tadayoshi Kohno. Genotype Extraction and False Relative Attacks: Security Risks to Third-Party Genetic Genealogy Services Beyond Identity Inference. In Network and Distributed System Security Symposium (NDSS). 2020.
Received: 2019-06-14. Accepted: 2019-10-07. Project URL: https://dnasec.cs.washington.edu/genetic-genealogy/.

Genotype Extraction and False Relative Attacks: Security Risks to Third-Party Genetic Genealogy Services Beyond Identity Inference

Peter Ney, Luis Ceze, Tadayoshi Kohno
Paul G. Allen School of Computer Science & Engineering
University of Washington
{neyp, luisceze, yoshi}@cs.washington.edu

***Abstract*—Customers of direct-to-consumer (DTC) genetic testing services routinely download their raw genetic data and give it to third-party companies that support additional features. One type of analysis, called genetic genealogy, uses genetic data and genealogical methods to find new relatives. While genetic genealogy is quite popular, it has raised new privacy concerns. Genetic genealogy services can be leveraged to find the person corresponding to anonymous genetic data and have been used dozens of times by law enforcement to solve crimes. We hypothesized that the open design and broad API offered by some genetic genealogy services raise other significant security and privacy issues. To test this hypothesis, we analyzed the security practices of GEDmatch, the largest third-party genetic genealogy service. Here, we experimentally show how the GEDmatch API is vulnerable to a number of attacks from an adversary that only uploads normally formatted genetic data files and runs standard queries. Using a small number of specifically designed files and queries, an attacker can extract a large percentage of the genetic markers from other users; 92% of markers can be extracted with 98% accuracy, including hundreds of medically sensitive markers. We also find that an adversary can construct genetic data files that falsely appear like relatives to other samples in the database; in certain situations, these false relatives can be used to make the re-identification of genetic data more difficult. These attacks are possible because of the rich set of features supported by the API, including detailed visualizations, that are meant to enhance usability. We conclude with security recommendations for genetic genealogy services.**

I. Introduction

At-home direct-to-consumer (DTC) genetic testing is now commonplace; more than 25 million people have taken DTC genetic tests and over 100 million are expected to be tested in the next few years [34]. The dramatic rise in DNA testing has given consumers unprecedented access to their genetic information so they can learn about their health, ancestry, and family history. A popular feature offered by DTC testing companies is to let customers download their raw genetic results into genetic data files (GDFs) so they can analyze their own data. This feature has created a demand for specialized third-party analysis companies to help users analyze these raw GDFs. Third-party services do not generate genetic data directly; rather, they offer databases and tools to store and analyze it. In this paper, we focus on third-party services that specialize at the intersection of genetics and genealogy, known as *genetic genealogy*.

The main goal of genetic genealogy is to predict new familial relations using genetic information. This technique, known as *relative matching*, relies on algorithms that can predict the relatedness of two individuals by comparing their two GDFs. Intuitively, these algorithms rely on the fact that contiguous DNA segments are passed from parents to their children in a predictable manner, and thus, the number and length of shared DNA segments between two individuals can be used to predict their relatedness. This pair-wise matching algorithm can be scaled to query all individuals in a large genetic database to find all potential relatives for a given individual and is reliable for relatives as distant as 3rd cousins [22].

The growing size of genetic genealogy databases has challenged assumptions about the inherent anonymity of genetic data because relative matching can be used to re-identify anonymous DNA samples. The largest and most prominent of these third-party services is GEDmatch, which maintains a database with over 1 million GDFs [32]. In April 2018, it was revealed that analysis on GEDmatch played a crucial role in identifying the suspected Golden State Killer [7], [15]. Since then, private companies, like Parabon Nanolabs, have created "genetic genealogy" units to identify potential suspects using GEDmatch; over 25 cold cases have been solved with relative matching on GEDmatch [19]. However, there are no technical restrictions preventing anyone from applying these same re-identification techniques to other genetic data or samples, including anonymous research subject data [3], [14]. Current estimates predict that a genetic genealogy database containing 2% of a target population will produce a 3rd cousin or closer relative for over 99% of individuals, the same relative degree used to identify the Golden State Killer [14].

To summarize, open-access genetic genealogy services contain large amounts of privacy sensitive genetic data, can be effectively used to identify the source of genetic samples, have been used extensively by law enforcement, and may soon

Network and Distributed Systems Security (NDSS) Symposium 2020
23-26 February 2020, San Diego, CA, USA
ISBN 1-891562-61-4
www.ndss-symposium.org

be large enough to identify individuals from entire countries. Therefore, it is essential that we understand the security practices of major genetic genealogy services.

A. Motivation

Open-access genetic genealogy services offer a rich set of analyses and have complex APIs. Most importantly, the genetic genealogy API allows users to upload and compare GDFs to other users in the genetic database to find relatives. In some services the API gives the user a high degree of flexibility to query specific GDFs. Ideally, a query would not reveal unintended information about the state of the database. However, we hypothesize that a feature rich API, like the one supported by GEDmatch, will be challenging to secure. Specifically, we suspect that such services may be vulnerable to maliciously crafted queries that leak private information or data uploads that poison the genetic database.

Here, we do a focused security analysis of the GEDmatch API to better understand how security threats might manifest in the genetic genealogy ecosystem. We find that GEDmatch has significant privacy and data integrity issues that compound existing identity inference concerns. These results highlight the tension between the feature richness of a genetic genealogy service and the potential for security and privacy issues.[1]

B. Overview

In this paper, we explore the possibility of attacks against GEDmatch from an adversary that only uses the standard API (i.e., uploads normally formatted GDFs and runs standard queries). We make the following contributions:

- In Section III we threat model an open-access genetic genealogy API using the design of GEDmatch as a guide. We discuss what types of attacks are possible and how certain design choices can contribute to security risks.
- In Sections VI and VII we show how the results returned from GDF comparisons (i.e., genetic coordinates and visualizations) can be used by adversaries to extract raw genetic markers of any targeted user in the GEDmatch database. This attack is sufficiently powerful that it could be used to steal large portions of the genetic database.
- In Section VIII, we discuss how an adversary might poison the genetic database with spoofed or falsified GDFs to create spurious relationships. We then show how these GDFs can be easily constructed and uploaded to GEDmatch.
- In Section IX, we discuss possible mitigations to these security issues and other recommendations for the genetic genealogy industry.

II. Background and Related Work

In this section we present background information on DTC genetic testing and relative matching. We then survey related work on genetic identity inference and genome privacy.

[1] An earlier version of this work appeared in [33].

```
#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
...
```

Fig. 1. An example DTC generated GDF. Each line corresponds to a single SNP and includes a SNP identifier, chromosome number, base position within the chromosome, and the genotype of the SNP.

A. Direct-to-Consumer Genetic Testing

DTC genetic testing is marketed directly to consumers without going through an intermediary, like a healthcare provider. The most popular type of test uses dense genotyping arrays that probe between 0.5-1 million genetic markers. DTC testing kits are inexpensive ($<$ \$100 USD) and are mailed to customers to be returned with a DNA sample (usually saliva). Customers use the DTC genetic tests to get information on their genetic ancestry, genealogy, and health. Users can also download files from DTC companies, called genetic data files (GDFs), that contain their raw genetic results.

GDFs contain an individual's genetic information at specific one-base-long positions in the human genome, which are known to vary within the human population. These positions are referred to as *single nucleotide polymorphisms* (SNPs). The possible DNA bases found in the human population at a particular SNP are known as *alleles*, and the specific bases a person has at a SNP constitutes that individual's *genotype*. The genotype of each SNP in the GDF contains two DNA bases because chromosomes come in pairs (one from each parent). GDFs are encoded in a simple ASCII format, with each SNP recorded on a separate line (see Figure 1). The SNPs are first sorted by chromosome and then by position within each chromosome.

Since the rise of DTC testing, customers have wanted to interpret their own genetic data using additional online third-party services. To upload data to third-party services, customers usually download their GDF 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 file formats (e.g., VCF).

B. Relative Matching

Relative matching algorithms rely on the fact that more closely 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 attempt 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 GDFs can be used to locate these matching segments because they contain the genotype of an individual at positions throughout the genome.

Except for close relatives, like siblings, or when individuals descend from a small number of people, it is typical for

each matching segment to be shared on only one of the two chromosomes in a pair. This is because one chromosome comes from the mother and the other from the father, and most people are related through one branch of their family. When a DNA segment matches on only one chromosome, it is called a *half-match* and on both a *full-match*.

C. Identity Inference of Genetic Data

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 relative matching can be combined with genealogy information, like family trees, to identify the source (person) of the genetic data. This approach is known as *identity inference* or *identity tracing* and is used by law enforcement to identify suspected criminals from unknown DNA samples [13], [14], [19].

To perform identity inference the genetic genealogist must have or be able to construct GDFs for an unknown target individual and have access to genealogy information to construct family trees. The genealogist then queries the third-party service with the DTC profile to find the target's relatives and uses the genealogical information to determine the identity of the target.

Recent work has demonstrated that anonymous genetic data in public datasets can be de-identified using GEDmatch [14]. The ability to de-identify DNA data using third-party services largely depends on the number of individuals — and, in particular, the number of relatives — already in the genetic database: identification is easier with more matches. Using a database that includes 1.28 million individuals, researchers estimated that 60% of all individuals would be able to identify a third cousin or closer, the same level of relationship that was used in the Golden State Killer case; a database covering just 2% of the population would suffice to find a third cousin for nearly every person [14], [25].

D. Genome Privacy

To our knowledge, we are the first to experimentally explore the specific attacks surfaced and studied in this paper. 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 these domains focused on privacy-respecting methods for processing genetic data [4], [24], [36]. The field has continued to expand, as captured by surveys such as Akgün, et al. [2], Mittos, Malin, and De Cristofaro [30], and Naveed, et al. [31]. 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 [13] in their survey. Work at this intersection exploits genetic facts to compromise privacy. 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 target's possible surname [21]. Other works, such as [12] and [14], 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. As early as 2009, Goodrich observed that cryptographic approaches for computing over genomic data would *not* prevent information leakage resulting from those computations [17]. Essentially, knowing the output of a computation over two DNA sequences (even if computed cryptographically), and knowing one of those two DNA sequences, can leak information about the other sequence.

III. Genetic Genealogy Attack Surfaces

Open-access genetic genealogy services, like GEDmatch, are feature rich and have complex APIs. In this section, we consider the security implications of GEDmatch's design. The GEDmatch API lets users interface with the genetic database to upload data and run over twenty different genetic genealogy queries. The API is typically accessed via a web interface, but an open source command line interface is also available [23].

A. Genetic Database

The core function of the service is to mediate access between users running relative matching queries and the sensitive, raw genetic data that needs to be stored and accessed to return results. The GDFs uploaded by users — referred to as *kits* on GEDmatch — are stored in a database and assigned a unique kit-identifier that references kits in subsequent queries. Each GDF is also associated with metadata like name and email address, so it is tied to an individual's identity. Since the genetic database stores large quantities of privacy sensitive GDFs, it is a high risk target; for example, individual GDFs contain medically relevant markers. If the entire database was stolen, an adversary might have significant and permanent identity inference capabilities.

The way the database is structured also has important security implications. In the case of GEDmatch, the kits appear to be compressed with a lossy compression scheme, likely applied when the GDF is uploaded (Section V). Intuitively, compression might lower the risk of data theft because the raw genotype data is not stored; however, it actually creates new security issues because it makes it significantly simpler to extract genetic markers from other users through repeated querying (Section VI and VII).

B. GDF Uploads

One of the most significant API functions is to let users upload new GDFs to populate the genetic database. GDF uploads are necessary for a user to find new relatives or make other comparisons. GEDmatch supports GDFs produced by the largest DTC testing companies, but will accept any file as long as it conforms to the GDF format with a sufficient number of SNPs. This fact is significant because there are no current mechanisms to ensure that the uploaded GDFs actually originated from a DTC service (see Section IX for a discussion of possible solutions). Law enforcement has already leveraged this capability to upload GDFs that came from DNA processed outside of the DTC ecosystem [15]. However, the ability to upload unauthenticated GDFs opens services to additional security issues. For example, an attacker can upload

falsified GDFs or use pathological GDFs in malicious queries to extract private genetic markers from other users. The lack of authentication also means that GDF uploads could be a vector for malware.

C. Relative Matching

A relative matching query, called a *one-to-many comparison* on GEDmatch, takes a kit identifier as input and returns a list of potential relative kits, sorted by relatedness. The details on the relative kits include name, contact information, kit-id, and properties of the match. This list can be expanded to view over 2,000 matches with standard accounts and 100,000 with premium [8]. This lets an attacker scrape large numbers of kit-ids and user information for use in subsequent attacks. Attackers can recursively run additional relative matching with each identified kit to scrape even more kit-ids. Overall, this design makes it quite easy for an attacker to identify large numbers of users if they have uploaded GDFs.

D. Direct Comparisons

The API also lets users make direct GDF-to-GDF comparisons, known as *one-to-one comparisons* on GEDmatch, to investigate potential relatives in detail; all that is required is the kit-ids of the two kits being compared. This is important because it lets an attacker target anyone with a known kit-id if there are vulnerabilities in direct comparisons. One-to-one comparisons return visualizations called chromosome paintings that show differences between the kits on each chromosome and the genetic coordinates flanking the shared segments. These results are common in genetic genealogy because they let users understand precisely how they are genetically related to potential relatives. The segment coordinates are often maintained by "power users" in large spreadsheets, which are helpful to understand the inter-relatedness of different matches [10]. In Section VI and VII we show how both the chromosome paintings and segments coordinates on GEDmatch leak too much information about the underlying SNPs being compared. Using adversarially crafted kits, we were able to extract the majority of SNPs from a targeted kit.

E. Additional Utilities

GEDmatch offers 18 additional tools to upload additional data or query the genetic database. Each of these is a possible attack vector that could have security implications. However, in this work, we limited our focus to the most common genetic genealogy features.

IV. Ethics and Responsible Disclosure

In the subsequent sections we describe a number of experiments we did to understand the feasibility of different attacks on GEDmatch service. We took great care to perform these experiments ethically and legally and to disclose any vulnerabilities we uncovered responsibly. We elaborate on the specific precautions we took below.

Since GEDmatch is a live service, we had to ensure that all our experiments were not only legal, but that they respected the privacy of GEDmatch users and had minimal impact to the GEDmatch service. The GEDmatch Terms-of-Service allows raw data uploads from artificial DNA kits as long as they are: (1) intended for research, and (2) not used to identify anyone in the GEDmatch database. We ensured that we complied with both terms.

Further, to protect the privacy of any individual (both GEDmatch users and non-users), we derived all DTC profiles 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 real individuals, 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). The "Research" designation meant that our kits would not appear in the matching results of other users. Furthermore, to avoid any risk of de-anonymizing the anonymous donors in the 1000 Genomes Project and OpenSNP datasets, we analyzed only DNA matches between the experimental kits we uploaded and did not view any matching results containing real GEDmatch users.

Our University IRB determined, through written review, that our research did not require IRB oversight because all data 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.

Vulnerabilities we uncovered have been disclosed to GEDmatch who is actively developing mitigations to these problems. We also reached to US government stakeholders before the paper's release.

V. Reverse Engineering One-to-One Comparisons on GEDmatch

In subsequent sections, we show how the results from matching queries can be used to steal data or poison the genetic database. Experimenting with these attacks first required significant reverse engineering of one-to-one comparisons on GEDmatch. Here, we describe our experimental setup and an overview of reverse engineering results. We leave a detailed description of our reverse engineering procedure to Appendix A.

We created a free GEDmatch user account that was used for all experimentation in the following sections. All analysis was done with custom Python scripts and standard libraries. We used the Python Imaging Library (PIL) to process visualization images. All experiments on GEDmatch were conducted between 01/16/19 and 06/14/19.

A. Artificial Kit Design

To generate kits for experimentation, we combined genetic data from the 1000 Genomes project and DTC data files from OpenSNP [1], [18] using a modified methodology that Erlich et al. [14] used in their study of identity inference attacks on GEDmatch. 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

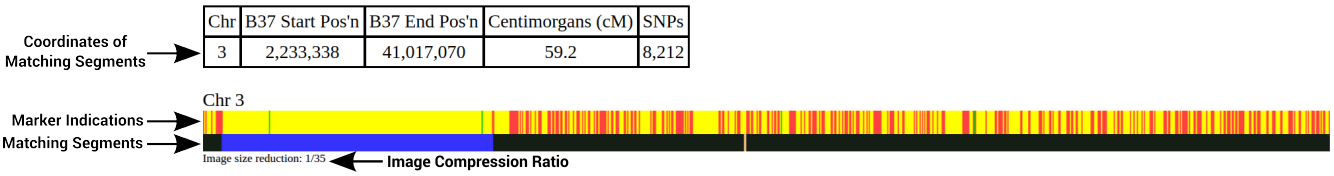

Fig. 2. One-to-one autosomal comparison results (default parameters) shown for chromosome 3. There is one matching segment between 2,233,338 - 41,017,070 (build hg37) with genetic distance 59.2 cM that includes 8212 SNPs. The image bars are compressed with at 1:35 ratio. On the marker indication bar (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.

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 to conform to the GEDmatch ToS.

We were primarily interested in understanding privacy risks to users that had their kits set to the default "Public" privacy setting on GEDmatch. This setting provides 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)[2]. 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 setting prevents kits from appearing in database-wide one-to-many queries but still lets the user run one-to-one comparisons if kit-ids are known.

B. Query Requirements

The precise SNPs in a GDF vary by the DTC company and chip version used to genotype the DNA sample. It is important that the adversary use 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.

When comparing two kits in a one-to-one comparison, both kit-ids are needed. Therefore, an adversary attempting to attack a specific target needs to know that target's kit-id. The kit-id of a specific individual can be found using a few methods:

- *Email via User Lookup Tools:* GEDmatch provides a "User Lookup" tool that can find a user's kit-ids via email address or genealogy ID (GEDCOM ID).
- *Relative Matching Queries:* As described in Section III-C, the results from relative matching queries can be used to scrape thousands of kit-ids.
- *Publicly Available Information:* A user may reveal a kit-id publicly, something we have seen numerous times on blog posts, Internet forums, and videos.

[2]In mid-May 2019, GEDmatch split the public setting into a public opt-in and public opt-out for kits related to law enforcement searches. This change does not impact our analysis.

C. One-to-One Comparison Details

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; see Appendix B for possible configurations. Each query returns a set of 22 matching results and comparison images, one for each of the autosomal chromosomes (see Figure 2 for chromosome 3 results). 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 a 1-dimensional pixel vector encoded in the GIF image file format. We were able to collect these 22 GIF images by downloading the web files containing the visualizations in the Chrome web browser. Four colors appeared 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 ignored that pixel in subsequent analysis.)

At full resolution, each pixel in the marker indication bar corresponds to a single SNP. This bases of a SNP in the first kit are compared to the bases of the same SNP in the second kit. If the genotype is the same in both kits at that SNP, then it generates a green pixel (full-match) in the marker indication bar; one base off and it produces a yellow pixel (half-match); and if both bases are different than it produces a red pixel (no-match). Moreover, the bases at each SNP are not compared in any order, since the genotyping process produces unordered data. However, this is further complicated by the fact that the genotype of each SNP appears to be compressed from 2-bit (i.e., {A,C,G,T}) to 1-bit (i.e., {0,1}) when making a comparison. Specifically, A/T are interpreted as one-bit (which we denote 0) and C/G as the other bit (which we denote 1). This means that A's are treated identically to T's and C's identically to G's when making a comparison, and so it is the 'one-bit' bases of the two kits that are being compared.

To make this concrete, consider an arbitrary SNP where the genotype of the first kit is AT and the second kit is CA.

First, the two genotypes are compressed to 1-bit (00 vs 10), and then the one-bit bases are compared, which results in a one base match and generates a yellow pixel.

We also uncovered the following facts about one-to-one comparisons that were necessary in our subsequent attacks.

- GEDmatch filters out most SNPs with a low *minor allele frequency* (MAF) — the frequency of the second most common SNP variant.
- GEDmatch ignores SNPs containing non-standard bases (like I/D or – –) in either of the compared kits.
- Only SNPs 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 marker indication pixel bar on any chromosome is displayed at a lower resolution if generates more than ˜32,000 pixels.

VI. Genetic Marker Extraction using Chromosome Visualizations

Here, we experimentally explore whether an adversary could use direct comparisons as an oracle to extract the DNA profile (raw SNPs) of another user of that service.

A. Generating Targets

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 V-A. 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.

B. 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 comparisons. This attack is broken up into a number of phases, which are described in detail 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. The resulting marker indication pixels leak too much information about the target's SNPs; an adversary can use the pixels to infer the underlying raw genotype of the target kit. However, recall that GEDmatch compares kits in a binary form, so in this case, the adversary can only extract the binary genotype of the target. In our experiments, we recovered the binary genotype for 61.0% of the SNPs in the target.

Next, the adversary can decompress the binary genotype into normal DNA bases using known allele frequency data. We were able to convert the binary genotype of the compressed SNPs 90.1% of the time. Finally, the adversary can fill in many of the missing SNPs using a well known genetic technique called *imputation*. After imputation, we extracted a total of 92.6% of the SNPs with 98.4% accuracy from our five targets.

C. Binary Genotype Extraction: Determining the Correspondence Between Pixels and SNPs

At full resolution, each marker indication pixel corresponds to the comparison of a single SNP. Therefore, if we know both which SNP corresponds to which pixel and a method to convert from pixel color to binary genotype, we can extract the binary genotype of each corresponding SNP. (See Figure 3 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 V that when two kits are compared, SNPs may be ignored for a number of reasons, including: the SNP is pre-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, e.g., a dash or I/D. In all 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`, which is a standard experimental kit except for three changes. First, all SNPs with a MAF less than 1% were removed. Second, 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). Finally, every other SNP was removed on chromosomes 1, 2, 3, and 6 to prevent more than 32,000 pixels from being generated, which lowers the marker indication bar resolution. The resulting kit contained 363,164 autosomal SNPs and resulted in 347,511 unfiltered SNPs when uploaded to GEDmatch, which left a substantial number of SNPs that were still being filtered.

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, consider the following: the binary genotype of the other kit is (1) homozygous (00 or 11), or (2) heterozygous (01). In case (1), every SNP in the unmodified `ext-kit` is 00 or 11, so 00/11 will be compared to 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 produces 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 SNPs.

We created a new kit, called `extmod(`n`)-kit`, which 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*.

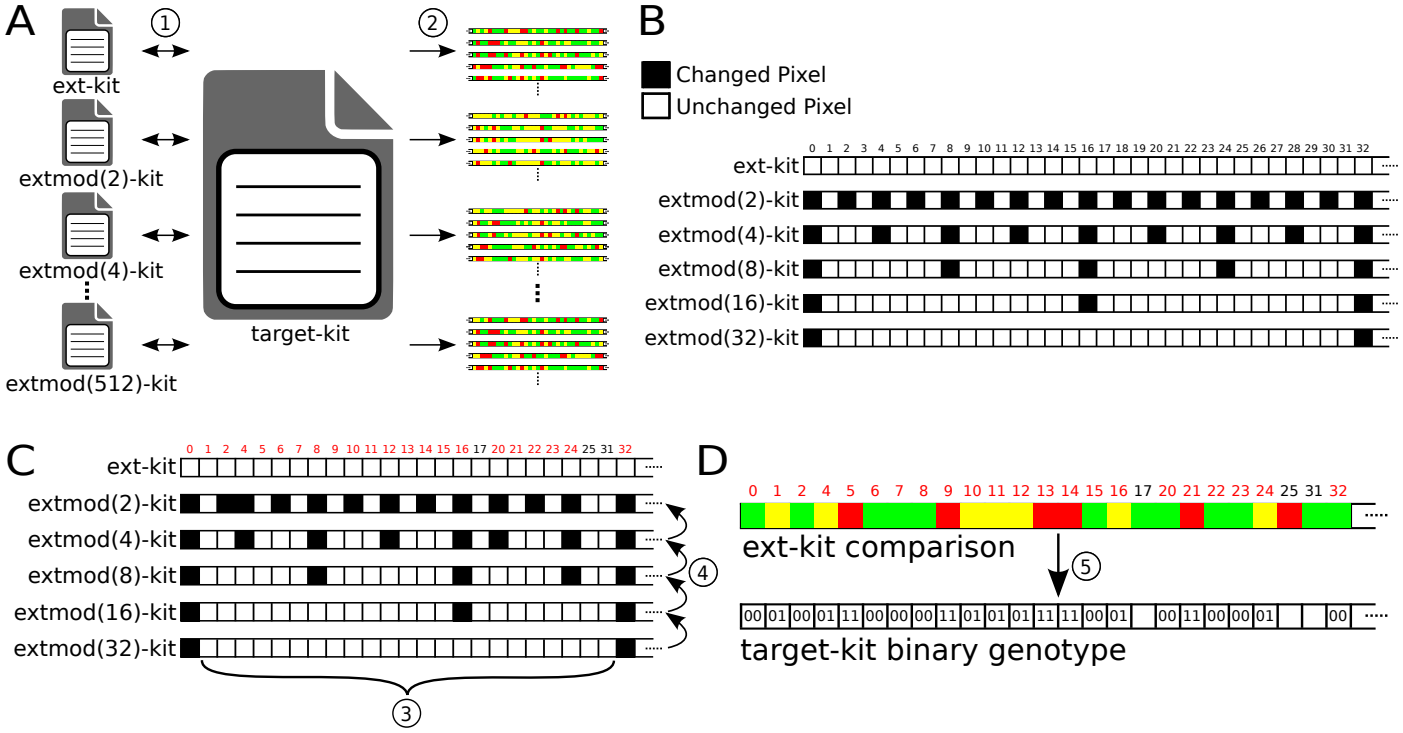

Fig. 3. **(A)**: One-to-one comparison of extraction kits to the target kit. **(B)**: Theoretical comparison when no SNPs are ignored. Changed pixels are at multiples of n. SNP indexes are listed above `ext-kit`. **(C)** Comparison when SNPs at indexes 3, 18, 19, and 26-30 are ignored. The correspondence could be determined for the SNPs at indexes shown in red. **(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.

At a high level, we can estimate the pixel-to-SNP correspondence for the changed pixel by counting the number of intervening pixels between the changed pixels. We can repeat this for differing values of n to find the correspondence for a large number of SNPs. Since the details of our method, while involved, are not critical to understanding the rest of this paper, we leave a detailed description of the pixel-to-SNP correspondence algorithm to Appendix D.

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(512)-kit`. Recall that when constructing `ext-kit`, half of the SNPs were removed on chromosomes 1, 2, 3, and 6 to ensure the marker indication pixel bar was at full resolution. Therefore, we had to repeat this procedure by constructing a different `ext-kit` where we alternated which SNPs were removed on chromosomes 1, 2, 3, and 6; we did this to find the correspondence of all SNPs on those chromosomes. In total, we constructed and uploaded 20 kits to GEDmatch.

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

D. 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` were compared to any other kit, we could 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 (the experiment 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. Similarly, 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 same method, we extracted the binary genotype of these SNPs. We predicted the binary genotype of all SNPs in `target(1)-kit` directly from the normal base-2 genotype and used this information to confirm that the binary genotypes from all of these SNPs were extracted correctly.

E. Decompress the Binary Genotype

Previously, 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. However, 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 were known to have only 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 has only two alleles that each correspond to different bits, then all binary genotypes can be decompressed. This situation is common because the least common single base pair mutations in the human population are A/T and C/G substitutions, which together account for only around 16.5% of all possible single base pair substitutions [39]. Such substitutions are the only ones that lead to ambiguous decompression; 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 decompressed each of these SNPs into the normal DNA bases and compared them to bases in the `target(1)-kit`, which confirmed that all these SNPs were predicted correctly.

Of the 9.9% of SNPs that 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 are thus not useful for comparisons.

F. Impute the Remaining SNPs

At this stage, we have inferred 337,468 of the 613,878 SNPs in the `target(1)-kit`. The last step is to predict the remaining SNPs in the target kit. To do this, we used a statistical technique called imputation which is designed to predict missing genotypes [28]. Imputation works more effectively when more data is 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 the `target(1)-kit` [29]; we pre-phased the SNPs with EAGLE2 and used the Haplotype Reference Consortium (r1.1) as the reference panel [27]. This imputed 231,126 additional SNPs with 96.0% accuracy.

G. Experiments with the Targets

To study the efficacy of this attack against targeted users, we extracted the five target kits from different accounts; 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 VI-C, VI-D, and VI-E), then predicted an additional 37.7% of the SNPs using imputation with 96.0% accuracy (Section VI-F). In total, we extracted an average of 92.6% of the SNPs with 98.4% accuracy.

After the extraction kits have been uploaded (a one time cost), the extraction takes around 10-20 seconds of comparison time on GEDmatch. Therefore, this attack could easily be scaled up to extract the genotype from large numbers of kits, which would only be limited by the kit identifiers that could be scraped.

Finally, we quantified the risk of genotype extraction to medically relevant SNPs. Using the ClinVar archive — a dataset that links human genetic variants and phenotypes — we selected all SNPs from the `target(1)-kit` that were categorized as either "pathogenic" or "likely pathogenic". The privacy risks were indeed significant: of the 608 medically relevant SNPs in the `target(1)-kit`, we were able to correctly extract 264 of these.

VII. Genetic Marker Extraction Using Matching Segments

We previously described how the marker indication bar can be used to extract the genotype of a large number of SNPs from some target kit. One defensive response might be simply to 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, leak enough information to extract specific SNPs of interest. (See Figure 2, and note the significantly lower resolution of the Matching Segments bar compared to the Marker Indications bar that we used in Section VI.)

Take matching DNA segments as an example. If an adversary can construct a matching segment around a specific SNP, then by changing the genotype of that SNP, the matching segment may split or disappear. This happens because the SNP will no longer match in the two kits, which cuts the matching segment in half. Therefore, an adversary may be able to use the presence or absence of a matching segment as an oracle to extract individual SNPs. Here, we show how this can be done on GEDmatch.

A. 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 matching segment 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, like minimum segment length, are configurable by the user when running a 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 VI-C for an explanation). Therefore, we can construct a matching segment in any given chromosome region by setting all SNPs in that region to AC.

B. 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. Assume a SNP of interest called 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 the region (S_{i-j},S_{i+j}) is at least 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 we must consider based on the genotype of S_i in the target: (1) S_i is missing or contains a non-standard allele, (2) S_i is 00 or 11, or (3) S_i is 01.

- *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 remain unchanged because the SNP is missing or it contains a non-standard allele. Therefore, this fact can be 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 entirely if the resulting smaller segments are below the minimum matching length. This will correspond to the extraction kit that 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 is set to 00, 01, and 11, then we know the target has genotype 01 because it is the only genotype that matches to all three kits.

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

C. 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 the `target(1)-kit`. To test the four possible situations, the target SNPs were 00, 01, 11, and the final one was missing entirely from the `target(1)-kit`. Each targeted SNP was on a separate chromosome so we could attempt to extract all at once.

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

VIII. Spoofed Relationships

In this section we explore our hypothesis that an attacker can upload spoofed or falsified GDFs to create spurious relative matches and forge familial relationships. We first demonstrate that an adversary can upload GDFs to GEDmatch that produce false relative matches and then consider how false relationships can be used maliciously.

A. Constructing False Relatives on GEDmatch

Recall that relative matching works by identifying matching segments between two GDFs, and the degree of relatedness is proportional to the total length of the matching segments. Resources like the Shared cM Project can help users find matching segment estimates for each type of relationship [6]. Suppose an adversary wants to spoof a 2nd cousin for some target on GEDmatch. The adversary can do this by uploading a GDF that produces the expected number and length of matching segments for a 2nd cousin; this can be extended to any desired relationship. An easy way to accomplish this is to replicating the matching segment coordinates of real relative pairs on GEDmatch — the adversary can view matches for any kit in the database if the kit ID is known. Since GEDmatch does not verify the identity of its users, the adversary is free to assign any metadata (e.g., name or email address) to the kit and user account.

From our investigation in Section V, we know that a SNP is half- or full-matching on GEDmatch if one or two of the compressed genotype bits match, respectively; a matching segment is a long run of half- or full-matching SNPs. An adversary can spoof a matching segment by copying one or two of the compressed bases from a run of SNPs in the target. Arbitrary matching segments can be spoofed whenever the binary genotype of the target kit is known; this can be found for any kit (with known kit-id) using the method described in Sections VI-C and VI-D. In the simple case when the adversary already has access to the target's GDF, they can copy SNPs directly from that GDF to duplicate matching segments.

We experimentally tested the above method to make a false child for the `target(1)-kit`. We first extracted the binary genotype of the `target(1)-kit` as usual and then set all SNPs in the false-child GDF to half-match the `target(1)-kit`; note, a parent-child relationship is expected to have half-matching segments on all chromosomes (approximately 3,400 cM of shared DNA). We uploaded the false relative GDF to GEDmatch and compared it to the `target(1)-kit`. This resulted in 3,411.1 cM of half matching segments and a most recent common ancestor estimate of 1 — the estimate expected for a parent-child relationship.

Our experiment shows that an adversary can create artificial half- and full-matching segments to spoof different relations; however, there are ways to make spoofed relatives appear more realistic, especially if the matches are scrutinized (as would be expected in a forensics investigation). Take the half-matching segments we constructed for the false child above. These were constructed by half-matching every SNP in `target(1)-kit`, however, in real parent-child relationships, half-matching segments will contain mixtures of half- and full-matching SNPs. The adversary could instead copy the distribution of half and full-matching SNPs from a real parent-child comparison to make a more realistic match.

In other cases the adversary may want to spoof a relative of a target who has other existing relatives in the database.

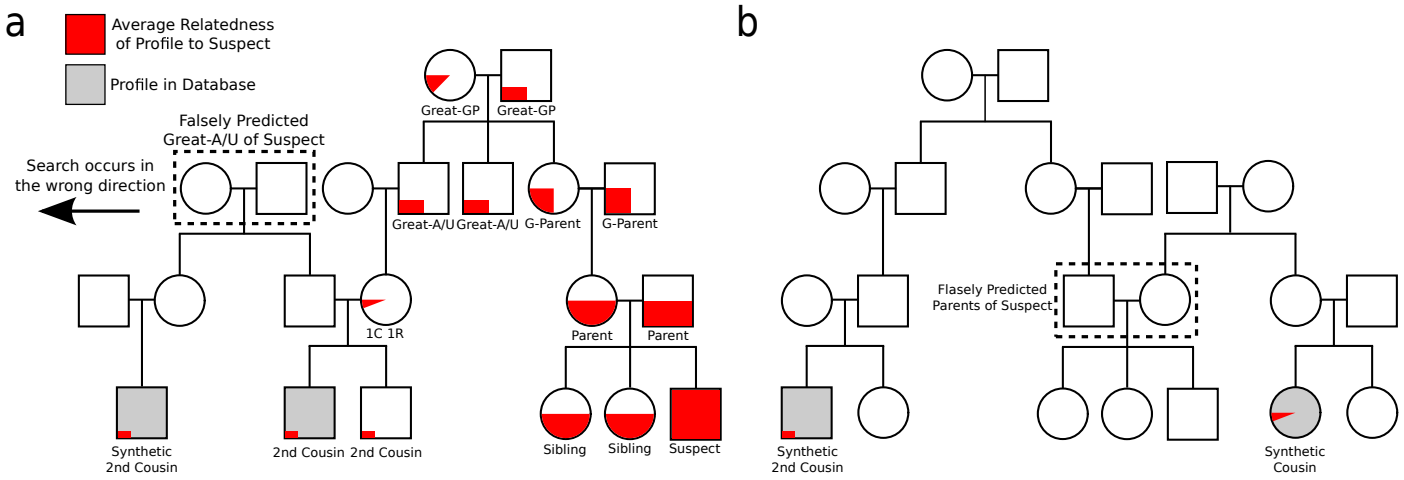

Fig. 4. Example attacks that disrupt genetic identify inference. **A**, An adversary wants to avoid identification when their 2nd cousin is already in a third-party database. The adversary uploads a falsified second cousin under the identity of a second individual that is related to the 2nd cousin but 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 that a couple was the adversary's parents.

Depending on where those existing relatives are in the family tree, the adversary may need the spoofed GDF to be related to both the target and the other existing relatives (i.e., two relatives of a target may be related to each other). In cases of genetic triangulation, the target, existing relatives, and the spoofed relative may need to match on some of the same segments. To solve both issues, the adversary can extract the binary genotype of the target's matches, in addition to the target. Then the spoofed GDF can be designed to have matching segments for both the target and the target's relatives.

B. Security Implications of False Relatives

Having confirmed that it is possible to spoof relations on GEDmatch, we now consider how an adversary can use this capability maliciously.

Commit fraud or harm reputation: The cultural and legal significance of family relations brings significant security implications. There are many reports of genetic genealogy users finding unexpected relative matches due to misidentified parentage or adoption [5], [9]; estimates put the non paternity rate in Switzerland and Germany at around 1% [35], [37]. Normal users are not likely to consider the possibility that matches can be spoofed, and therefore, may believe that an unexpected, spoofed match is legitimate. An adversary could use this capability to gain the trust of the victim or damage their reputation.

Method: The adversary identifies a target and then generates and uploads a spoofed descendent of the target to GEDmatch using the procedure described above. From the target's perspective, the spoofed relative will appear like an authentic relative match.

Disrupt genetic identity inference: To identify the person corresponding to an unknown genetic sample, a genetic genealogist will first find all the relative matches for the unknown sample and work through genealogy records to generate a large set of potentially thousands of possible identities. The genealogist will sift through these identities using the expected demographic information (e.g., age, gender, location, and ethnicity) and genetic triangulation to narrow the set of possible identities to a manageable number [19]. The adversary's goal is to use spoofed relations to disrupt or prevent this type of analysis. Note that when identity inference is used in a law enforcement context, the possible identities are treated as leads that are re-tested with traditional tests, like short tandem repeat fingerprinting [19]; therefore, the adversary can only prevent discovery.

Method: It is difficult to experiment with attacks against genetic identity inference in an ethical manner, especially in the forensics context, due to the unique nature of each search — which is a manual and expert driven process. Below we describe a number of theoretical attacks that may disrupt or make identity inference more challenging. However, we emphasize that further research is needed to confirm the effectiveness of these attacks in real searches.

Recall that falsified relatives can be uploaded under falsified or fictitious identities on GEDmatch. In Figure 4 we show two simplified scenarios where an adversary implies a false identity inference using spoofed relative under an assumed identity. Figure 4(A) depicts a scenario where an adversary wants to avoid identity inference with a second cousin already in the database. The adversary uploads a spoofed second cousin under the identity of an individual that *is not* related to the adversary but *is* related to the second cousin. This falsely implies that the adversary is on a different branch of the family tree. Note, that to be consistent with the topology of the family tree, the spoofed second cousin should be a second cousin of the adversary and a first cousin of the true second cousin already in the database. In a second attack, shown in Figure 4(B), the adversary uploads spoofed relatives to different branches of a family tree to falsely imply that the adversary is a descendent of an unrelated couple.

There are other circumstances that can make identification more challenging. For example, international genealogy records can be hard to find [19]; therefore, an adversary could upload spoofed matches under the identity of individuals

that are located internationally or deceased. People from endogamous populations — those that are highly interrelated — can be harder to identify because they typically have a high number of matches and complex family trees. Companies like Parabon Nanolabs believe an individual is likely from an endogamous population if they have at least ten, 70cM or greater matches (3rd cousin or closer) [19]. An adversary could leverage this fact by uploading a large number of distant spoofed matches, similar to what is expected in an endogamous population, to make the search space intractable. Finally, if the adversary controls the identity of the spoofed matches (e.g., email address or other contact information) then they could be contacted by investigators — as was done with a second cousin in the Golden State Killer search — which would alert the adversary that they are being actively searched [16].

IX. Discussion

Third-party genetic genealogy analysis has been a useful tool for millions of customers; however, as this study demonstrates, vulnerabilities in these services can raise significant security and privacy risks. Here, we reflect on these issues to make a number of security recommendations and discuss how these attacks might generalize to other services.

A. Genetic Data File Authentication

All of the attacks we identified were possible because there are no technical restrictions preventing an adversary from uploading falsified or pathologically designed GDFs. Therefore, we strongly suggest that relative matching queries and direct comparisons be restricted to data that was generated by a DTC testing company, which we call DTC-authentication. Erlich et al., proposed a possible DTC-authentication scheme to prevent unauthorized identity inference [14]. They proposed that DTC testing companies digitally sign GDFs and include the signature in the GDF header so the file can be verified by third-parties.

Such a scheme would also be effective at preventing the vulnerabilities we uncovered because an adversary could not make comparisons with arbitrary data, which was required to extract genetic markers, or generate falsified relatives. We further suggest that the genotyping instrument itself digitally sign the data it generates so it can be traced to a single instrument, company, and time. This DTC-authentication scheme is also flexible because it gives third-party services the control to decide when to verify GDFs and can allow for exemptions like approved law enforcement use.

B. API Fixes and Security Mitigations

First, we suggest that all direct comparisons require that one of the GDFs was uploaded by the user. This way a user cannot compare arbitrary GDFs to one another only using GDF identifiers. Moreover, we suggest limiting direct comparisons to GDFs with a minimum degree of relatedness. This will restrict the possible set of GDFs an adversary can target but does not significantly affect usability because unrelated GDFs are rarely compared.

As a second defense-in-depth strategy, we suggest that third-party services rate limit queries, especially against the same GDF, because repeated querying was necessary to extract raw SNPs. Moreover, services should consider implementing some form of anomaly detection; both the artificial GDFs and query results from our experiments were highly anomalous and should be detectable with a simple classifier.

Services should also consider implementing deterrent measures as well. For example, services could alert users whenever their kits are queried by someone else in a direct comparison, which could give advanced notice of an attack.

Chromosome visualizations and other details returned from matching queries are important features to help users understand how they are related to each other, and so we do not recommend eliminating these features entirely. However, services should be very wary of returning fine grained visualizations and precise genetic coordinates because they may leak unintentional information. System designers will have to find a balance between the precision of matching results with the possibility of data leakage.

Finally, the data storage method and matching algorithm should be scrutinized. In the case of GEDmatch, genotype compression during comparisons made it much simpler to construct half- and full-matching segments. This design may also contribute to other risks we did not explore, like denial-of-service attacks, because an adversary may be able to construct pathological kits that match all GDFs in the genetic database.

C. Generalizability of Attacks

This work focused on a security analysis of GEDmatch, but an important question is how these results might generalize to other genetic genealogy services. GEDmatch is somewhat unique because of its size (largest third-party genetic genealogy service), the breadth of its API, and its prominent use in criminal forensics. However, many of the features, like relative matching, chromosome visualizations and segment coordinates in results, and raw GDF uploads are common among third-party and DTC services.

The other most significant third-party genetic genealogy service is an academic research effort called DNA.Land[3], which maintains a genetic database with over 160,000 GDFs [38]. Similar to GEDmatch, DNA.Land supports relative matching that returns chromosome visualizations and precise matching segment coordinates [11]. However, DNA.Land has a much more restrictive API than GEDmatch, which we suspect makes marker extraction attacks much more difficult in practice.

DNA.Land does not support direct comparisons between arbitrary kits, so it is more difficult to target specific users, and the resolution of the chromosome visualizations is much lower, which obfuscates SNP level details. The matching coordinates appear to be high resolution, so single SNP extraction (like that shown in Section VII) may still be possible. However, DNA.Land does significant preprocessing of each kit before analysis, including imputation, and uses a different matching technique based on the GERMLINE algorithm, which further complicates these attacks [20], [38]. Thus, we think the risk from marker extraction is much less severe on DNA.Land

[3]DNA.Land is currently transitioning from an academic to commercial service, which requires all previously collected user information and GDFs be deleted.

because of its significantly restricted API compared to GEDmatch and different algorithms used in comparisons.

Without DTC-authentication, it will be difficult to prevent false relative GDFs from being uploaded to any service as long as the attacker has access to a target's genetic sample or GDF. Therefore, an adversary should still capable of uploading false relatives to DNA.Land. However, this attack is much more severe on GEDmatch than DNA.Land because the attacker can first extract the target's genotype, and thus, target anyone in the database. It may also be possible that vulnerabilities in another service, like GEDmatch, could help bootstrap relative spoofing attacks on DNA.Land because the attacker could gather more information about the target, like the underlying genotype, using the vulnerabilities in the other service.

To summarize, while we did not experiment with DNA.land, we believe the more restrictive API and algorithmic differences makes DNA.Land significantly less vulnerable to the security issues raised in this paper. Further, if security issues were to manifest, they would be much harder to target and scale. This highlights how variations in APIs and other design choices can lead to significant differences in security, but that improved security may come at the cost of reduced functionality to users. Further, we suspect that feature-rich APIs provide value to the genetic genealogy community, and might contribute in part to GEDmatch's popularity. Hence, should new services emerge, we conjecture that they may face security challenges if they also provide broad APIs that appear favored by customers.

X. Conclusion

In this paper, we explored new threats to genetic genealogy beyond identity inference attacks. As this work demonstrates, genetic genealogy services can be difficult to secure because of their open nature and the rich set of features they support. We hope this contributes to a discussion in the computer security and broader genetics community about emerging security risks to genetic genealogy services and spurs future research on the secure design of genetic genealogy systems, especially as these services continue to be used in high-stakes applications, like criminal forensics.

Acknowledgements

This research was supported in part by the University of Washington Tech Policy Lab, which receives support from: the William and Flora Hewlett Foundation, the John D. and Catherine T. MacArthur Foundation, Microsoft, and the Pierre and Pamela Omidyar Fund at the Silicon Valley Community Foundation. It was also supported by a grant from the DARPA Molecular Informatics Program. We thank Ryan Calo, Bill Covington, and Elena Ponte from the UW Law School for giving us legal insights on drafts of this paper. We thank Franziska Roesner from the UW Security and Privacy Lab for early feedback on the paper and Sandy Kaplan for editing advice. We thank Stefan Katzenbeisser for shepherding this paper, and all our reviewers for insightful comments and feedback, all of which helped improve this paper.

References

[1] 1000 Genomes Project Consortium and others, "A global reference for human genetic variation," *Nature*, vol. 526, no. 7571, p. 68, 2015.

[2] M. Akgün, A. O. Bayrak, B. Ozer, and M. S. Sağiroğlu, "Privacy preserving processing of genomic data: A survey," *Journal of Biomedical Informatics*, vol. 56, pp. 103–111, 2015.

[3] P. Aldhous, "We tried to find 10 BuzzFeed employees just like cops did for the Golden State Killer," https://www.buzzfeednews.com/article/peteraldhous/golden-state-killer-dna-experiment-genetic-genealogy.

[4] M. J. Atallah, F. Kerschbaum, and W. Du, "Secure and private sequence comparisons," in *WPES*, 2003.

[5] A. 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.

[6] B. T. Bettinger, "The Shared cM Project 3.0 tool v4," https://dnapainter.com/tools/sharedcmv4.

[7] K. V. Brown, "DNA website had unwitting role in Golden State manhunt," Bloomberg, 2018.

[8] L. Coakley, "Tips for using GEDmatch," http://www.genie1.com.au/blog/78-tips-for-using-gedmatch, 2016.

[9] ——, "DNA success stories," http://genie1.com.au/blog/80-dna-success-stories, 2018.

[10] K. Cooper, "Taking it to the next level — DNA spreadsheets," https://blog.kittycooper.com/2016/09/taking-it-to-the-next-level-dna-spreadsheets/.

[11] "Relative finder information," https://dna.land/relative-finder-info.

[12] P. M. Ellenbogen and A. Narayanan, "Identification of anonymous DNA using genealogical triangulation," bioRxiv, Tech. Rep., 2019.

[13] Y. Erlich and A. Narayanan, "Routes for breaching and protecting genetic privacy," *Nature Reviews Genetics*, vol. 15, no. 6, p. 409, 2014.

[14] Y. Erlich, T. Shor, I. Pe'er, and S. Carmi, "Identity inference of genomic data using long-range familial searches," *Science*, vol. 362, no. 6415, pp. 690–694, 2018.

[15] T. Fuller, "How a genealogy site led to the front door of the Golden State Killer suspect," New York Times, 2018.

[16] M. Gafni, "The woman behind the scenes who helped capture the Golden State Killer," The Mercury News, 2018.

[17] M. T. Goodrich, "The mastermind attack on genomic data," in *IEEE Symposium on Security and Privacy*, 2009.

[18] B. Greshake, P. E. Bayer, H. Rausch, and J. Reda, "OpenSNP–a crowdsourced web resource for personal genomics," *PLoS One*, vol. 9, no. 3, p. e89204, 2014.

[19] E. Greytak and C. Moore, "Closing cases with a single SNP array: Integrated genetic genealogy, DNA phenotyping, and kinship analyses," International Symposium on Human Identification, 2018. http://docs.parabon.com/pub/Parabon_Snapshot_Scientific_Poster-ISHI_2018.pdf.

[20] A. Gusev, J. K. Lowe, M. Stoffel, M. J. Daly, D. Altshuler, J. L. Breslow, J. M. Friedman, and I. Pe'er, "Whole population, genome-wide mapping of hidden relatedness," *Genome research*, 2009.

[21] M. Gymrek, A. L. McGuire, D. Golan, E. Halperin, and Y. Erlich, "Identifying personal genomes by surname inference," *Science*, vol. 339, pp. 321–324, 2013.

[22] B. M. Henn, L. Hon, J. M. Macpherson, N. Eriksson, S. Saxonov, I. Pe'er, and J. L. Mountain, "Cryptic distant relatives are common in both isolated and cosmopolitan genetic samples," *PLOS ONE*, vol. 7, no. 4, pp. 1–13, 04 2012.

[23] N. Homer, "Gedmatch tools," https://github.com/nh13/gedmatch-tools.

[24] S. Jha, L. Kruger, and V. Shmatikov, "Towards practical privacy for genomic computation," in *IEEE Symposium on Security and Privacy*, 2008.

[25] J. Jouvenal, "To find alleged Golden State Killer, investigators first found his great-great-great-grandparents," Washington Post, 2018.

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

[27] P.-R. Loh et al., "Reference-based phasing using the haplotype reference consortium panel," *Nature Genetics*, vol. 48, no. 11, p. 1443, 2016.

[28] J. Marchini and B. Howie, "Genotype imputation for genome-wide association studies," *Nature Reviews Genetics*, vol. 11, no. 7, p. 499, 2010.

[29] S. McCarthy et al., "A reference panel of 64,976 haplotypes for genotype imputation," *Nature Genetics*, vol. 48, no. 10, p. 1279, 2016.

[30] A. Mittos, B. Malin, and E. D. Cristofaro, "Systematizing genome privacy research: A privacy-enhancing technologies perspective," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 1, pp. 87 – 107, 2019.

[31] M. Naveed, E. Ayday, E. W. Clayton, J. Fellay, C. A. Gunter, J.-P. Hubaux, B. A. Malin, and X. Wang, "Privacy in the genomic era," *ACM Computing Surveys*, vol. 48, no. 1, pp. 6:1–6:44, 2015.

[32] S. C. Nelson, D. J. Bowen, and S. M. Fullerton, "Third-party genetic interpretation tools: A mixed-methods study of consumer motivation and behavior," *The American Journal of Human Genetics*, 2019.

[33] P. M. Ney, "Securing the future of biotechnology: A study of emerging bio-cyber security threats to DNA-information systems," Ph.D. dissertation, University of Washington, April 2019.

[34] A. Regalado, "More than 26 million people have taken an at-home ancestry test," MIT Technology Review, 2019.

[35] G. Sasse, H. Müller, R. Chakraborty, and J. Ott, "Estimating the frequency of nonpaternity in Switzerland," *Human Heredity*, vol. 44, no. 6, pp. 337–343, 1994.

[36] D. Szadja, M. Pohl, J. Owen, and B. Lawson, "Toward a practical data privacy scheme for a distributed implementation of the Smith-Waterman genome sequence comparison algorithm," in *NDSS*, 2006.

[37] M. Wolf, J. Musch, J. Enczmann, and J. Fischer, "Estimating the prevalence of nonpaternity in Germany," *Human Nature*, vol. 23, no. 2, pp. 208–217, 2012.

[38] J. Yuan, A. Gordon, D. Speyer, R. Aufrichtig, D. Zielinski, J. Pickrell, and Y. Erlich, "DNA.Land is a framework to collect genomes and phenomes in the era of abundant genetic information," *Nature genetics*, vol. 50, no. 2, p. 160, 2018.

[39] Z. Zhao and E. Boerwinkle, "Neighboring-nucleotide effects on single nucleotide polymorphisms: A study of 2.6 million polymorphisms across the human genome," *Genome Research*, vol. 12, no. 11, pp. 1679–1686, 2002.

APPENDIX A
REVERSE ENGINEERING ONE-TO-ONE COMPARISONS

We began our investigation of one-to-one comparisons by running a comparison between two kits. See Table I for a description of the kits we used during experimentation.

To construct these two kits, which we denote as `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; other times, we copied both bases to replicate full-matches (recall that each SNP has two bases, one from each parent).

A. 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

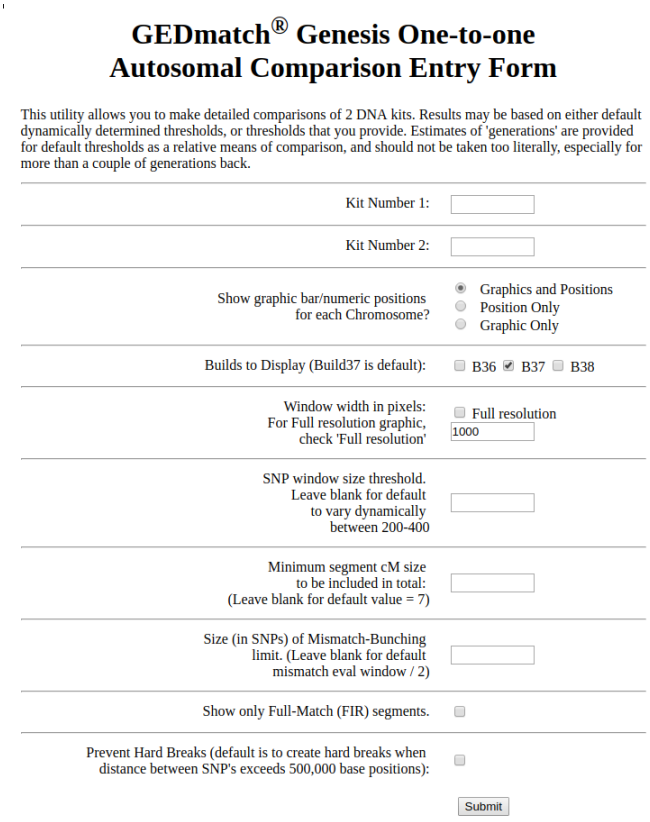

Fig. 5. Screenshot of GEDmatch's web form for one-to-one autosomal comparisons.

referred to simply as 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; therefore, 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 MAF [26]. To test this hypothesis, we used 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. 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.

B. Additional Comparison Details

This describes additional experiments that were necessary to fully understand GEDmatch's one-to-one comparisons.

- *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 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.

C. 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. This let us 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 treat 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, there is a match (green), only one the same (yellow), or both different (red). Therefore, the color is determined by counting 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. It is unclear whether GEDmatch actually stores the genotype of each SNP in this binary encoding or whether the binary encoding is computed from normal genotype data when making comparisons; however, it is not necessary to know what is stored for our attacks to be successful.

Appendix B
GEDmatch Screenshots

Figure 5 provides a screenshot of the GEDmatch interface when requesting one-to-one autosomal comparisons.

Appendix C
Matching Pseudocode

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

```
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
```

Appendix D
Pixel-to-SNP Correspondence Algorithm

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 would be m pixels for that chromosome, and the pixel at index i would correspond with S_i. Similarly, the pixels at indexes that are multiples of n (i.e., $0, n, 2n, ...$) will be the changed pixels (see Figure 3B). However, since some of the SNPs are ignored, the indexes of the changed pixels will shift (Figure 3C).

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:

$$k = j + \left\lceil \frac{g+1}{n} \right\rceil \times n \quad (1)$$

A. Proof of Above Equality

To prove the lower bound, we know that $k > j$ because the corresponding pixel for k is at a higher 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. If $r < n$ (i.e., fewer than n SNPs were filtered), then $(k-j-1)-g = r < n$. g can then be written as $g = k-j-1-r$

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

B. 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 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. (A boundary case must 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 1, this will hold as long as fewer than n SNPs are missing before S_x; otherwise, x will be a lower bound on the corresponding SNP index.)

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 I. CLASSES OF KITS USED IN GEDMATCH EXPERIMENTS.

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 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 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.

C. 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 the `ext-kit` with AC. If the `extmod(`$\frac{n}{2}$`)-kit` is compared to the same kit as the `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 the `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 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.