Control of Sensitive Data in Systems with Novel Functionality

Joseph Anthony Calandrino

A Dissertation
Presented to the Faculty
of Princeton University
in Candidacy for the Degree
of Doctor of Philosophy

Recommended for Acceptance
by the Department of
Computer Science
Adviser: Edward W. Felten

June 2012

Abstract

Advances in computer science have enabled analysis of data in ways previously unthinkable. This has led to powerful new uses of data, providing us with countless benefits across virtually all aspects of our lives. For systems utilizing sensitive data, novel functionality has sometimes provided novel routes for exposure of the underlying data. This functionality may come with dangerous new assumptions or undermine old ones, allowing unexpected inferences. As a result, release of seemingly innocuous information may reveal sensitive data in surprising new ways. This exposure can be detrimental, but it can often enable desirable new capabilities as well. Regardless of whether these exposures can help or harm us, we benefit from a deeper understanding of when and how they can arise.

We explore this issue in the context of three systems. First, we examine the impact on election systems of recent advances in the ability to reidentify sheets of paper. These advances pose some threat to the secret ballot, but they enable new measures for verifying election integrity. Next, we develop techniques for and discuss the use of markings on Scantron-style bubble forms as a biometric. These forms are used in a variety of circumstances, and potential implications vary from enabling cheating detection on standardized tests to undermining anonymous surveys. Finally, we examine data leakage from collaborative filtering recommender systems, finding that recommendations can be inverted to infer individuals' underlying transactions. This demonstrates that even subjecting data to massive-scale aggregation and complex algorithms can be insufficient to protect sensitive details. By explicitly considering the ways in which novel functionality exposes sensitive data, we hope to reduce the risk for those whose intimate details may be encoded in this data while encouraging responsible uses of the data.

Acknowledgements

I thank my advisor Ed Felten for the opportunity to attend Princeton and the support he has shown in academic and non-academic matters. My research group has contributed hard work, insights, and friendship to my graduate experience. For that, I thank Will Clarkson, Ian Davey, Ari Feldman, Shirley Gaw, Alex Halderman, Josh Kroll, Tim Lee, Harlan Yu, and especially Bill Zeller. Bill entered in my graduate class and provided technical expertise, solidarity during difficult classes, and plenty of humor. More importantly, he was a kind and loyal friend. With his passing, the world has lost not just a brilliant computer scientist but also a wonderful person. He is deeply missed.

From CITP, I thank David Robinson, Steve Schultze, Laura Cummings-Abdo, Jeff Dwoskin (who provided this handy thesis template), Joe Hall, and Nick Jones for their ideas and support. In the Department of Computer Science, Melissa Lawson has always been there to lend a helping hand, ensuring that everything from my admission to my FPO went smoothly. Beyond Princeton, Arvind Narayanan, Ann Kilzer, and Vitaly Shmatikov were superb research collaborators. Over summers, Lea Kissner, Paul Mattal, Dave Wagner, and Shabsi Walfish gave valuable guidance. Alf Weaver, Paul Reynolds, Nina Mishra, and Dave Evans were wonderful mentors during my time as an undergraduate and graduate student at the University of Virginia. For supporting my graduate education, I am grateful to the Department of Homeland Security Fellowship Program and Google.

My committee provided valuable advice during the thesis process. For many of them, their support extends through my classes, generals, papers, and pre-FPO to today. For that, I thank Andrew Appel, Brian Kernighan, Jen Rexford, Dave Walker, and Ed. Special thanks to Andrew, Jen, and Ed for reading this whole thing.

I thank my family. Mom, Dad, James, John, Big Grandma, Little Grandma, Aunt Sylvia, Uncle Jimmy, Erica, Eva, and many others have been there when I needed

them, routinely putting my interests before their own. Suffice it to say that a proper acknowledgment would extend beyond the length of this thesis. Finally, I thank those that helped me through a difficult period late in my graduate career. Many are already listed, but Alison Boden, Amy Lin, Rachel Lin, Erin Mishkin, Amber Oliva, Amy Ostrander, and Ashley Thrall also deserve my most heartfelt gratitude. In particular, I thank Amy Lin for showing the good in the world during a hard time.

For Bill Zeller.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The digital world is increasingly defined by us, its inhabitants, as opposed to dedicated content providers and architects. Today, we do not simply read books and newspapers, watch movies, or listen to recordings. We produce data and influence how it is organized, connected, and presented. Various sites serve as repositories for bits created by or about us, hosting and sharing everything from our home videos to scans of our election ballots. Tools like recommender systems and search engines exploit patterns mined from our activity to uncover subtle preferences and personalize our experiences. Computer scientists continually develop powerful new tools for analyzing user data, but the output of those tools nevertheless remains a product of user data. While the churn of innovation may rapidly render cutting-edge technologies antiquated, the surge of ideas built on leveraging data related to individuals appears less ephemeral. These new technologies can improve our lives, but they can also expose our personal data in novel, unexpected ways.

Creative uses of data about us have provided us with countless benefits. Researchers use this data to study everything from the spread of epidemics to the efficacy of government programs. Engineers and entrepreneurs use our data to build new systems and businesses. Analysts and marketers apply data about us to refine

experiences ranging from web browsing to offline grocery shopping. Regardless of the specific application, this data allows others to better understand the intricacies of our world and to better address our unique traits and tastes. These benefits only increase with advances in algorithms and computer hardware, which improve our ability to dissect and exploit data. We cannot predict all of the novel future uses of data by or about us, but many such uses will undoubtedly make our lives better and easier.

Often, novel uses of data are accompanied by novel assumptions. Our imaginations limit our ability to predict how others may analyze the output of systems using either today's techniques or future technologies. As a result, we may fail to anticipate the information revealed and connections enabled by certain uses of data. This failure may lead us not only to forgo advantageous capabilities but also to overlook sensitive details of our lives exposed when seemingly innocuous information is released. Beneficial surprises are not inherently more likely than detrimental ones. Consequently, novel functionality also tends to enable novel routes for data leakage unless care is taken to identify and prevent potential leaks. Whether revealed data can be put to positive or negative uses, we benefit from an awareness of these potential uses.

This dissertation examines the phenomenon of unexpected data byproducts exposed by novel functionality in the context of three systems. All three systems are sometimes used with data that either is intrinsically sensitive or stems from sensitive user actions. The impact of inadvertent disclosure of this data varies by circumstance, so we explore the implications in the context of each system.

Voting systems provide an interesting lens for studying this phenomenon due to their unique goals. An ideal voting process would instill well-founded confidence in an election's outcome and protect voter privacy while being easy and efficient for all stakeholders. Changes in technology both inside and outside of the voting booth influence these properties, and the impact of changes on the secret ballot may be particularly subtle and unexpected. Details ranging from procedures to physical

properties of election artifacts can influence ballot secrecy. We focus on the impact of recently uncovered techniques for reidentifying sheets of paper based on physical properties like surface texture. As we discuss, the application of these techniques to physical ballots can enable positive results for election verification purposes but can also provide an undesirable peek at voter's choices.

Second, we consider systems incorporating Scantron-style bubble forms. These forms are used in a wide variety of contexts where the identity of the person completing the form is critical. For example, the name written on a standardized test should match the person completing it. Conversely, anonymous surveys should effectively sever the connection between a surveyed individual and her responses. Surprisingly, we discover that a person's superficially generic markings on these forms can serve as a weak biometric. Because this biometric links people and forms, the positive and negative implications of these findings extend to the variety of contexts involving bubble form use.

Finally, we explore a more indirect use of sensitive data in the context of collaborative filtering recommender systems. These systems use past actions of individuals to generate recommendations. Use of these systems has become widespread on commercial web sites. We show that recommender systems can unexpectedly reveal fine-grained details of the actions that underlie recommendations, potentially without user knowledge or consent. In this case, the revealed data is aggregated over thousands to millions of people and no longer uniquely connects to any single one of them. In spite of the challenges that these circumstances pose for an attacker, we show that the impact of a single person's actions on the system's output may expose an association between the person and those actions.

1.1 Contributions

This dissertation makes three primary contributions:

- *Election ballots:* We show that existing paper fingerprinting techniques can threaten the secret ballot and enable new election verification techniques. The quintessential democratic election involves voters privately completing paper ballots and dropping them into a locked ballot box. Although new technology has been introduced to the voting booth, security concerns have continued to emphasize the value of a verifiable paper trail. Blank sheets of paper may look identical, but recent research has undermined this belief. Fiber patterns and other characteristics allow reidentification of individual sheets using inexpensive, widely available tools [23]. We explore the implications of paper fingerprinting on elections, discussing threats, mitigation approaches, and newly enabled fraud- and error-detection measures. This work was performed in collaboration with William Clarkson and Edward Felten. It was presented at the 2009 Electronic Voting Technology Workshop/Workshop on Trustworthy Elections (EVT/WOTE '09).

- *Fill-in-the-bubble forms:* We discover that the markings individuals make on Scantron-style fill-in-the-bubble forms serve as a weak biometric. Use of these forms historically came with an implicit assumption that the forms are identifying only if the answers or other information explicitly provided on them are identifying. Using a unique combination of image-processing and machine-learning techniques, we find that the visible physical markings made by individuals on these forms can themselves be identifying. Using these markings, we can identify an individual's bubble form uniquely in a set of more than 90 sample surveys with accuracy above 50%. These techniques can also identify an impostor completing a form on another person's behalf, obtaining simultaneous false

positive and false negative rates below 10% on our sample data. This research was joint work with William Clarkson and Edward Felten and was presented at the 20th USENIX Security Symposium (USENIX Security 2011).

- *Collaborative filtering recommender systems:* We demonstrate that changes in recommendations produced by collaborative filtering recommender systems can expose users' actions. These recommender systems use patterns derived from past user actions to suggest items. Individual actions may be sensitive, but recommendations are a product of aggregation over massive datasets. Although recommendations are not directly connected to individual users, we show that changes in recommendations can reveal sensitive data. A user's single action can yield changes that expose both the action and the responsible person. In a test on the recommender system for the site Hunch, our custom algorithms infer a third of test users' choices with no errors. This project was joint work with Ann Kilzer, Arvind Narayanan, Edward Felten, and Vitaly Shmatikov. It was presented at the IEEE Symposium on Security and Privacy 2011 (Oakland 2011).

1.2 Organization

The remainder of this dissertation is organized as follows. Chapter 2 provides the reader with some background in this subject area, focusing on sensitive data leaks from supposedly benign datasets and previously proposed mitigation techniques. In Chapter 3, we discuss elections and the impact that technological advances have on ballot secrecy and other aspects of the voting process. Chapter 4 explores the potential use of markings on fill-in-the-bubble forms as a weak biometric and the consequences of our results. Chapter 5 examines collaborative filtering recommender systems, considering the ability to infer individual user transactions from recommen-

dations that are aggregate byproducts of large-scale transaction datasets. Chapter 6 concludes and discusses the implications of this work for current and future privacy and security research, particularly work on privacy-enhancing technologies.

Chapter 2

Background

We begin by exploring previous cases in which researchers extracted sensitive details from data that appeared innocuous. In these cases, a trusted party possessed records documenting intimate details of people's lives. Those details ranged from medical history to search queries. Given the research value of this data, the party wanted to release the records. Unfortunately, public exposure of such records could harm the corresponding people. Seeking to resolve this tension, the trusted party tried to scrub the data of any link from the records to the people, supposedly "de-identifying" or "anonymizing" the data. These scrubbing methods proved insufficient, however, allowing others to reconstruct a link.

Concerns regarding accidental data leaks in these and related scenarios have led researchers to pursue a variety of mitigation strategies. Development of these strategies has refined our formal definitions of privacy and enabled stronger theoretical privacy guarantees. Existing techniques have a number of limitations, but research on this topic has led to critical advances. We hope that refinement of these techniques and future advances will provide stronger defenses against scenarios like those in this dissertation.

2.1 Inadvertent Exposure from Scrubbed Data

Previous work in this area has focused on connecting people with records supposedly scrubbed of identifying information. These cases are examples of a more general problem: inference of sensitive details from released data. Released data may be "scrubbed" records, but it also may be the result of complex aggregate functions computed over volumes of sensitive data.

Massachusetts medical records. In the mid-1990's, a government agency in Massachusetts wished to release the medical records of state employees to interested researchers [82]. Recognizing the sensitive nature of this data and wishing to protect patient privacy, the agency decided to remove explicit identifiers such as patient names from the records. This approach won the public support of then-Governor William Weld. To analyze the degree of anonymity afforded by this de-identification approach, Latanya Sweeney—then a doctoral student at MIT—purchased the voter registration list for the city of Cambridge, Massachusetts.

Critically, both the employee medical records and the voter records retained each corresponding person's gender, date of birth, and zip code. Unlike the medical records, however, the voter registration list also contained the voter's name [82]. Sweeney's later work suggested that gender, birth date, and zip code could be sufficient to uniquely identify up to 87% of the population [95]. These attributes alone proved sufficient to identify not only the voter registration record for Governor Weld—a Cambridge resident—but also his medical record, linking the two uniquely. Sweeney provided the governor's records to his office to demonstrate the issue [82].

Sweeney's work highlights the difficulty of removing identifying information from records. Even when a well-intentioned organization makes a legitimate effort to remove identifying information, these efforts may prove insufficient. Unexpectedly distinguishing features or unforeseen auxiliary data can undermine the best intentions.

As a result, numerous groups have encountered similar issues to those faced by Massachusetts.

AOL search records. In 2006, AOL released nearly twenty-million search queries made by 657,000 users over three months with the goal of assisting academic researchers [10]. This dataset associated all queries made by a user over a three-month period with a numeric identifier representing the user. While the search queries ranged from the mundane to the crude or disturbing, the dataset did not contain AOL screen names or other explicitly identifying information linking the queries to users. Unfortunately, the queries themselves proved to be identifying, revealing details like searcher location and age. Two New York Times reporters used the search queries for user 4417749 to narrow the set of possible AOL customers to Thelma Arnold of Lilburn, Georgia. In response to the danger that this dataset posed to user privacy, AOL stopped distributing it [10].

Netflix Prize dataset. Several months after AOL's ill-fated data release, Netflix decided to conduct a one-million-dollar contest seeking improvements to its recommender system [76]. Netflix released a large ratings dataset to contestants. This dataset contained over 100,000,000 movie ratings made by approximately 480,000 customers over six years, with each rating accompanied by a date. In an attempt to resolve privacy issues, Netflix created this dataset by sampling from its full dataset, removing explicitly identifying information, and purportedly perturbing the data slightly by adding noise. Within three weeks of the dataset's release, Arvind Narayanan and Vitaly Shmatikov documented methods for using this dataset to learn potentially sensitive details about Netflix customers [75] (later expanded to [76]).

As in the previous cases, Narayanan and Shmatikov demonstrated that auxiliary data—partial prior knowledge about a user's movie-watching habits or public IMDb reviews—enabled re-association of Netflix customers with full sets of ratings [76].

This occurred in spite of noise added to the data and was feasible even with limited or imperfect auxiliary knowledge. As a result, the dataset could potentially allow inferences such as political affiliation, sexuality, or religious beliefs. Narayanan and Shmatikov went a step further by providing methods enabling probabilistic inferences. While earlier work focused on exact links between people and records, Narayanan and Shmatikov argued that unique identification is unnecessary to reach damaging probabilistic conclusions [76].

Additional cases. While we consider only three prominent examples of sensitive data inadvertently exposed by superficially innocuous information, additional cases exist. For example, a line of work focuses on removing sensitive details from network data and the difficulties in doing so [83, 28, 26, 27]. Similar cases cover a variety of scenarios, including fingerprinting physical devices over a network [58], inferring language and phrases from encrypted VoIP calls [101, 100], extracting information from encrypted video [90], and demonstrating the difficulty of anonymizing social network graphs and data [9, 77].

Our work. Our work differs from this previous work in several respects. Unlike our primary examples, Chapter 3 and Chapter 4 focus on cases in which technological changes undermine long-held beliefs regarding the anonymity offered by a system. This differs from a failure to locate and remove all potentially identifying information from a dataset at the time of its release. In these cases, we also strive to uncover any positive implications of the exposed data. Our work in Chapter 5 extends our analysis of data leakage to complex aggregate data produced by recommender systems. Because user records are not published in this case—just aggregate results over those records—previous approaches for recreating links between records and individuals are not applicable. Instead, we are forced to infer details more indirectly. Together, this work explores how novel functionality can expose sensitive data in unexpected ways.

2.2 Existing Mitigation Approaches

The traditional approach for ensuring data confidentiality is to encrypt the data. Unfortunately, naive encryption of data prior to its release would destroy the data's utility. Consequently, we begin by considering the strengths and weaknesses of traditional cryptography before discussing alternative mitigation approaches. As we discuss, a recently developed approach known as differential privacy provides theoretical leakage guarantees that resemble cryptographic approaches while striving to minimize changes to the data.

Classic cryptography. Perhaps the most well-known use of cryptography is to encrypt a message such that a recipient with a key can decrypt the message but an eavesdropper without the key cannot learn anything. These systems typically come with a proof that deriving meaningful information from the encrypted text is equivalent to solving a difficult mathematical problem.

The guarantees that we desire when scrubbing data differ dramatically from the guarantees provided by this prototypical cryptosystem. While the formal theoretical rigor underlying cryptography's guarantees is desirable, basic encryption presents a binary choice: hide all data—sensitive or not—by releasing an encrypted dataset or release all data unencrypted. We desire an alternative that removes sensitive data while preserving some of the data's utility.

Secure multi-party computation bears some resemblance to our needs but is not a perfect match. These systems provide a method for parties to compute a function that depends on input from each party without requiring any party to reveal their input to others [102]. This does not ensure that the results of the functions themselves do not betray sensitive data, however. For example, Chapter 5 explores the potentially sensitive data revealed by the output of recommender systems. Because our algorithms rely only on the output of the recommender systems, the recommender

system's method of obtaining input has no impact on the effectiveness of our inference algorithms. We require an approach that not only protects a system's input from direct disclosure but also scrubs output of sensitive details.

Secure multi-party computation reveals a distinction in the type of data revealed. In cases like the Massachusetts, AOL, and Netflix data leaks above, the release comprised a full dataset supposedly scrubbed of identifying details. In other cases, an entity computes and reveals the results of queries over sensitive data. These queries may be predefined or chosen (potentially dynamically) by the recipient, and the results must be scrubbed of sensitive details. Some overlap exists between the full-dataset and query cases: a de-identified dataset could be constructed using a scrubbed query for each record in the dataset, and scrubbed queries could be computed directly on a scrubbed dataset. An ideal approach would handle both cases.

Query Auditing. When responding to queries over sensitive data, an attractive approach is to audit query responses and deny ones that would leak sensitive data precisely. For example, suppose that a system provides the average salary of Alice and Bob. If a future query is for the average salary of Alice, Bob, and Carol, an adversary could combine the answers to learn Carol's exact salary. One technique for avoiding these cases is to audit queries and refuse to answer ones that directly leak sensitive data [31].

This approach is less effective when probabilistic inferences may be an issue. Even if nobody knows whether a specific member of a group has an expensive-to-insure illness, that member's eligibility for an insurance plan may hinge on aggregate group figures. An additional drawback—which applies to any system for removing sensitive data from query responses—is that past queries may prevent the system from ever responding to important future queries. Finally, this approach may be NP-hard

depending on the specific scenario [22], and refusals to answer queries can also leak information if not performed cautiously [56].

k-anonymity. In our example cases from the previous section, the party releasing data relied on removal of explicitly identifying information to separate records from individuals. This approach fails when combinations of remaining fields continue to uniquely identify individuals, particularly when auxiliary information ties those combinations of fields back to explicitly identifying information. The case of Massachusetts medical records saliently demonstrates this pitfall. To mitigate this issue, Samarati and Sweeney proposed the notion of k-anonymity [89], which others have refined [66, 61]. k-anonymity—which is both a definition of privacy and a technique for achieving it—ensures that any record in a dataset matches at least $k-1$ other records in the dataset on any identifying combinations of metadata fields. To do so, it suppresses or generalizes those fields.

Consider the earlier medical records example. To protect Governor Weld's record, we could remove the day of the month on which he was born, merge several adjacent ZIP codes into a single group, and suppress gender. These changes should ensure that his record falls into a bucket of at least k records. The resulting buckets allow computation of broader aggregate statistics but permit individuals to hide in a larger group. This bears some resemblance to early query auditing techniques in that it helps prevent unique identification of an individual.

k-anonymity fails to prevent problematic inferences completely, however. While we can adjust k to put individuals in larger or smaller buckets, suppose that all medical records in Governor Weld's group demonstrate a risk for either heart disease or cancer. Even without identifying his record directly via metadata, we have learned a great deal of sensitive information [66]. Effectively, any deviation of a group's aggregate attributes from general population aggregates can be problematic [61]. In

addition, auxiliary data can continue to cause issues when applied to non-metadata attributes. For example, if an individual is known to have undergone knee surgery and only a single medical record notes such a procedure, that person's medical record can be inferred [66]. Finally, k-anonymity has difficulty retaining useful information when handling high-dimensional data [2].

Addition of noise and differential privacy. An alternative technique for protecting privacy is to add noise to the data such that aggregate statistics remain but the original data has been masked. This idea has yielded a number of approaches, perhaps the most well-known of which is differential privacy.

The past decade has seen a significant quantity of research studying the limits of and techniques for adding noise to datasets or queries on datasets [3, 30, 38, 12, 4]. This work has led to a formal definition of privacy that Cynthia Dwork has termed differential privacy [33, 35]. Suppose that a trusted party has access to a statistical database of records containing sensitive values. That party wishes to output a function over the dataset while preventing inferences—exact or probabilistic—about any individual's sensitive details. To help hide these details, the party adds random noise to the function's results, resulting in a random function, F. If datasets D and D' differ on at most a single record and $S \subseteq \mathit{range}(F)$, then F provides ε-differential privacy if [33, 35, 36]:

$$\frac{\Pr[F(D) \in S]}{\Pr[F(D') \in S]} \leq \exp(\varepsilon) \tag{2.1}$$

At a high level, this means that no single record should significantly influence the function's output: the probability that F's output falls in any range should differ only marginally with a change in any one record. Note that F does not need to be a single query. It could technically be a request for the entire dataset or the transcript of a lengthy sequence of queries [36].

Differential privacy is used as an umbrella term referring to both the definition and methods for achieving it. Those methods depend on the exact use of the data. Dwork, McSherry, Nissim, and Smith have suggested adding noise chosen from the Laplace distribution to each dimension of the result (*i.e.*, each query permitted) [36]. The standard deviation of this noise would depend on the impact that any single record can have on the original function's output.

The significance of differential privacy is based perhaps more on the definition than methods for achieving it, however. While a different definition might be necessary under certain cases, it is the first formal definition of privacy to see widespread acceptance.

To minimize the noise added, existing differential privacy methods require fairly narrow limits on either the queries permitted or the types of data accepted. Imagine that an adversary repeatedly attempts to isolate the value of a certain record in the dataset through a series of queries. Eventually, the noise can be averaged out unless either the adversary is cut off or the responses are extremely noisy.

Though this limitation may not be insurmountable with clever approaches, it poses a challenge for algorithms like collaborative filtering recommender systems. Creating recommendations for each movie, book, or other item available from a website can require a massive number of queries over each sensitive, high-dimensional transaction record in a customer dataset. Making matters worse, the need to periodically update recommendations means that this process may repeat indefinitely.

We explore the application and limitations of differential privacy, k-anonymity, and other mitigation techniques in Chapter 4 and Chapter 5. Chapter 6 discusses useful avenues for future work.

Chapter 3

Some Consequences of Paper Fingerprinting for Elections

If elections are a cornerstone of democracy, few systems are more important than voting systems. Although technology has gradually made its way into the voting booth, paper records continue to play a pivotal role in many voting systems. Paper is cheap, familiar, and reliable; and paper records can be read and written by people and machines. Furthermore, paper records can provide an audit trail to protect against security or reliability issues in other aspects of the system. As a result of paper's positive properties, the voting systems most widely recommended today by election security experts rely on keeping paper records of each ballot.

Even when using paper, however, care must be taken to ensure election integrity and ballot secrecy. Over the past several years, advances in algorithms and computer hardware have enabled paper fingerprinting—unique identification of individual sheets of paper based on physical characteristics [71, 15, 104, 23]. State-of-the-art techniques can identify a piece of paper without any need to mark or alter it in advance and require no more than a commodity desktop scanner and personal computer. The ability to uniquely identify a sheet of paper presents both drawbacks and benefits for

voting systems that rely on paper ballots. In certain scenarios, tracking of individual paper ballots could undermine ballot secrecy. In others, it can enable more efficient auditing techniques. We discuss the positive and negative implications of existing paper fingerprinting techniques on elections in this chapter, exploring various threat scenarios and proposing an auditing scheme that relies on fingerprinting individual ballots. Through vigilance and careful procedures, election officials may mitigate many threats posed by paper fingerprinting while harnessing its benefits.

Traditional paper-based voting systems, optical scan voting systems, and DRE-VVPAT systems[1] yield paper ballots containing the voters' choices for various contests. Suppose that someone has access to the paper ballots (or scans of the ballots) before and after an election. If this person can identify the paper ballot you will use to vote, she can reidentify and recover the paper ballot containing your votes when election day concludes.[2]

Recent advances have made this threat more realistic, showing that it is possible to identify sheets of paper based on unique physical characteristics. These characteristics are often imperceptible to the human eye yet accurately measured by a machine. The most advanced systems measure slight variations in color or 3D surface texture of paper, requiring only a commodity desktop scanner and custom software. This is accomplished without modifying the paper in any way. Fingerprinting of paper ballots presents additional privacy challenges that must be addressed by election officials to ensure ballot secrecy. Fortunately, many attacks based on fingerprinting are nontrivial; they require access to the paper ballots and certain equipment at particular times. We survey the risks that fingerprinting poses based on details of

[1]Direct recording electronic voting machines with a voter-verified (or voter-verifiable) paper trail—a computerized voting system producing redundant paper ballot records that each voter may verify and approve. Security analyses of computerized voting systems have exposed numerous vulnerabilities that could compromise the integrity of elections performed using computerized systems without a paper trail (for example, see [57, 39] and references therein), emphasizing the value of such a trail.

[2]Some fingerprinting techniques can reidentify paper even if the voting process results in markings or creases.

common voting systems and the malicious party's level of access. Based on common themes, we provide suggestions for election officials to minimize these risks.

The ability to uniquely reidentify paper ballots also has implications for election auditing. Audits of paper ballots following an election are critical for verifying the outcome, helping to detect possible errors or fraud. When performing an audit, we may select individual ballots for review based on fingerprints, enabling ballot-based auditing. As we discuss, ballot-based auditing—selection and verification of individual ballots—can be far more efficient than existing widely used techniques. Fingerprints effectively serve as serial numbers on the ballots without posing the same privacy risks as serial numbers. Paper fingerprints may also help uncover attacks that are problematic for some auditing schemes, including attacks that rely on ballot box stuffing. These additional checks can potentially result in greater election integrity. Because the security of the auditing proposals relies on the fingerprinting process, we briefly discuss the security of paper fingerprinting.

The remainder of this chapter is organized as follows. Section 3.1 provides background information on the topic of paper fingerprinting as it is relates to fingerprinting paper ballots. Section 3.2 describes new threats posed by paper fingerprinting as well as choices that may mitigate these threats. Section 3.3 proposes new auditing techniques that make use of paper fingerprinting. Finally, Section 3.4 concludes.

3.1 Fingerprinting Background

A ballot fingerprint is a set of physical properties of a paper ballot that enable reidentification of that ballot. In the past few years, several systems capable of identifying pieces of paper based on difficult-to-reproduce physical characteristics have been developed [71, 15, 104, 23]. These fingerprinting systems can rely on any distinguishing characteristics of paper. For example, when viewed up close, the surface of a sheet of

paper is not perfectly flat, but actually a tangled mat of paper fibers. This nonuniformity in a paper's surface texture (the 3D shape of a paper's surface) is created during the manufacturing process and is unique to each sheet of paper. The FiberFingerprint system of Metois et al. first introduced the notion of using surface texture to uniquely identify a document [71]. FiberFingerprint uses a custom device to measure "inhomogeneities in the substrate" of a document, from which a unique identifier is derived. Laser Surface Authentication is a more advanced technique that measures a document's surface texture using a high-powered laser microscope [15]. Other methods, such as Print Signatures, can identify sheets of paper based on different unique characteristics, such as the random ink splatter that occurs around the edges of any features printed on it [104].

Recent work by Clarkson et al. describes another method for identifying sheets of paper based on surface texture [23]. Clarkson's method uses only a commodity scanner to reconstruct a paper's surface texture, from which a secure and robust fingerprint is derived. This technique requires no costly or specialized equipment, relying only on an unmodified commodity desktop scanner and custom software, and produces fingerprints that are verifiable using any appropriate-resolution scanner. In addition, the process does not require modification of the paper in any way, leaving no evidence that a sheet was fingerprinted.

During an election, paper ballots may be treated harshly. These ballots may be creased or folded, and they are modified by markings indicating voter choices. Any paper identification system must be robust to harsh treatment and moderate levels of marking. We focus on the methods developed in [23] because this system utilizes inexpensive, widely available equipment and can reidentify a document even when a sheet of paper is handled harshly or modified by being printed or scribbled on. We briefly review two variants of the system from [23], one requiring only a single scan and another more secure version that requires multiple scans. We then discuss

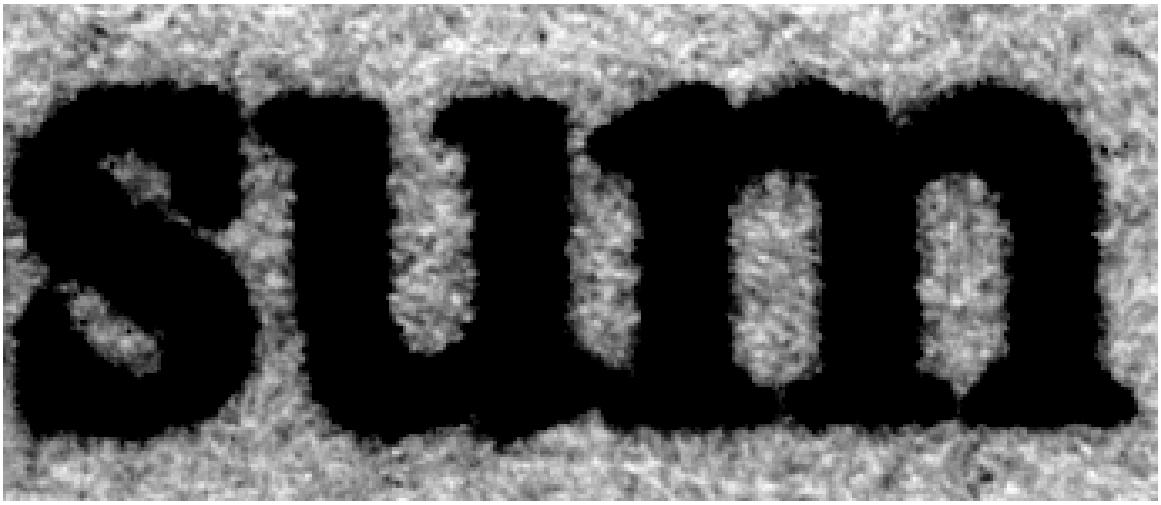
a method for protecting the privacy of paper ballot fingerprints. For more details, please refer to [23].

Single Scan. This scheme identifies a sheet of paper based on slight variations in its color. After scanning a sheet of paper at a high resolution—at least 1200 DPI—and adjusting the contrast we can see slight variations in color due to surface texture. See Figure 3.1(a). A fingerprint for the sheet of paper is derived from these variations. However, the resulting fingerprint may not be highly secure against forgery, as high quality printers may be able to mimic the measured feature.

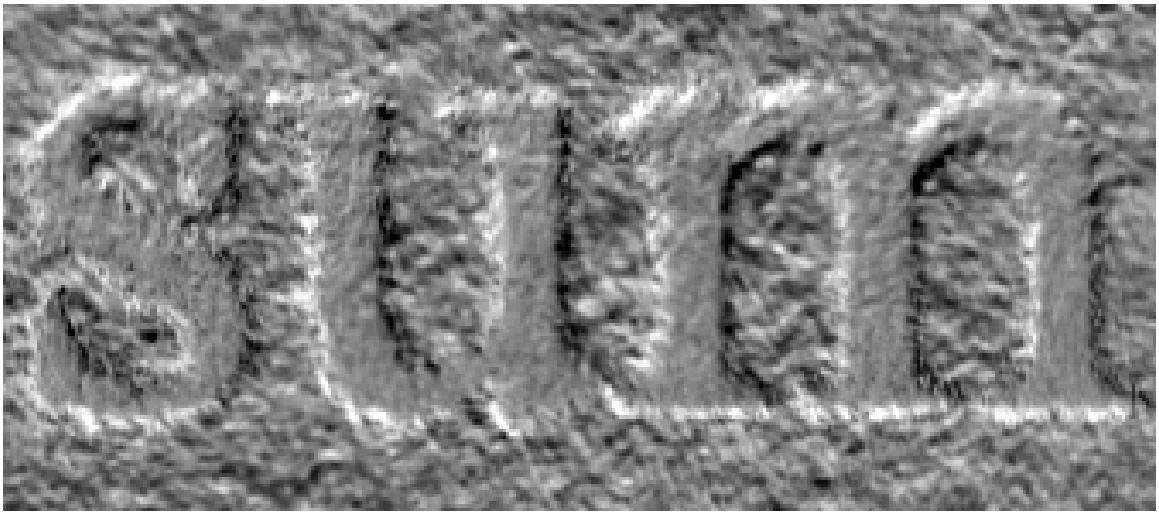
Multiple Scans. This more secure version reconstructs the surface texture of a sheet of paper by taking multiple scans at different orientations. This is accomplished without changing the paper in any way, using only an unmodified commodity scanner and custom software [23]. A sheet's surface texture is estimated by scanning the document at four orientations and combining the data inferred from each scan. The level of detail is dramatic, see Figure 3.1(b). After measuring the surface texture, a concise fingerprint for the document is generated. The fingerprint under this variant is secure to forgery, as the surface texture of paper is difficult to duplicate or precisely modify. The resulting document fingerprint is also robust, allowing successful re-identification even if the document is creased, scribbled on, soaked in water, or printed on [23].

Protecting Ballot Fingerprints. Regardless of which variant is used to generate a ballot's fingerprint, knowledge of the raw fingerprint could aid an adversary in generating a counterfeit sheet. If ballot fingerprints are published, then the privacy of each fingerprint must be protected, using a secure sketch [32] or some other method.

A secure sketch protects the fingerprint from accidental disclosure and only allows verification when provided with a document close to the original. Intuitively a secure



(a) A contrast-adjusted 1200 DPI scan of a sheet of paper. In the background, we see slight color variations, which are used to identify individual sheets of paper. Actual size: "sum".



(b) Image depicting the surface texture as measured by two 1200 DPI scans. After only two scans, the geometry of the paper surface is clearly visible. The embossing effect of the text is actually the wax ink sitting on top of the paper surface.

Figure 3.1: Example paper scans.

sketch is a fuzzy hash function, with similar (nearby according to an appropriate metric) inputs producing identical outputs. Similar to cryptographically secure hash functions, given an output it is infeasible to find an input that produces identical output. For ballots, the verification process of a fingerprint would compare the fuzzy hash of a ballot's fingerprint against a list of candidates. This enables the secure verification of ballot fingerprints. As Dodis et al. demonstrate, a ballot's secure

sketch would not reveal the underlying fingerprint, and thus hides the ballot's surface texture [32]. In addition, fingerprinting ballots using the methods described in [23] will not reveal the votes that these ballots contain. This permits the publication of a list of secure sketches of ballot fingerprints, without revealing the fingerprints themselves. This list can later be used to verify that a ballot is legitimate (among other things) by comparing its fingerprint to those on the list.

3.2 New Threats

Although the ability to fingerprint and reidentify paper ballots poses a serious threat to voter privacy, officials may take steps to mitigate these threats. In this section, we detail several threats to voting systems incorporating paper ballots and discuss possible countermeasures.

Even without fingerprinting techniques, a number of methods exist for making paper ballots unique and potentially identifiable. Voters may choose an unusual write-in candidate or a unique combination of candidates to create a distinctive ballot. Given enough races, a pseudorandom selection of choices would with high probability create a distinctive ballot that could be later identified. Alternatively, poll workers may mark ballots with invisible ink or lightly crease the corner of a ballot. The paper fingerprinting system in [23] differs from past ballot tracing techniques (including invisible ink) because there is no way to detect, even by close inspection, whether a ballot is traceable. Every ballot is inherently traceable.

We consider only threats created or made easier by fingerprinting. For example, suppose that an optical scan system stores a fingerprint or high resolution scan of every ballot in the order they are cast. Given this information from the voting machine's memory, an observer that watches when voters submit their ballots and can later examine the ballots for fingerprints could reidentify a voter's paper ballot,

thus revealing the voters' choices. Given access to the voting machine's memory, an easier and equally devastating attack exists without fingerprinting, however: the optical scan machine could simply store all votes cast in order, and the attacker could match the ordered list of voters against the ordered list of votes. Therefore, this threat is not considered.

The threats that we do consider are those in which someone other than a voter can learn the voter's choices for various contests. This may be with the consent of the voter. To sell her vote, a voter may provide the fingerprint of her paper ballot to a purchaser. If paper ballots are revealed on or after election day, the purchaser can reidentify the appropriate ballot and verify the choices. Alternatively, someone may want the ability to uncover nonconsenting voters' choices—whether due to curiosity or a desire to coerce voters—by fingerprinting blank paper ballots in advance and later associating voters with completed ballots. Anyone from voters and election officials to paper mill workers may be a participant in these threat scenarios. We use the term adversary to refer to any party that seeks to undermine ballot secrecy.

This reveals two general classes of attacks relating to paper identification. The first occurs when an adversary is able to fingerprint a significant portion of the ballot stock prior to an election and later associate voters with particular ballots. The second occurs when an individual voter scans her own ballot, either voluntarily or under coercion. We now discuss threats for various voting systems under each attack model.

3.2.1 Ballot Stock Fingerprinting

In this attack model, an adversary with access to the ballot stock is able to fingerprint or make high-quality scans of a significant portion of the ballots. By combining the ballot fingerprints with information about the order of voters an adversary may be able to undermine ballot secrecy. We discuss this attack for various voting systems.

DRE-VVPAT with Paper Spool. Paper fingerprinting poses a serious threat to DRE-VVPAT systems using printers with paper spools. We do not consider continuous spool-to-spool DRE-VVPAT systems, as they fundamentally fail to protect the secret ballot [55].[3] In cut-and-drop VVPAT systems, the record tape is used in a fixed order. As each voter casts his ballot, the tape is cut, dropping the segments into a box. Suppose that an adversary has unmonitored access to this paper spool prior to election day. The adversary can unroll the spool and repeatedly fingerprint short segments of the paper tape in order, storing the fingerprints in that order. He can then re-roll the spool and return it. Because the DRE will use the paper tape in order, the order of the segment fingerprints will correspond to the order that ballots are cast on the DRE. Although some segments may be destroyed when the ballots are separated, use of short enough segments can ensure that at least one segment remains intact per paper ballot, allowing complete re-ordering. For this attack to succeed, an adversary would need the ability to scan paper tape segments prior to the election, to observe the order that some or all voters enter the voting booth, and to fingerprint some or all of the resulting paper ballots later.

It is important to note that an adversary need not observe all voters or fingerprint all ballots before and after the election. Suppose that an adversary can reidentify her own ballot without scanning it, perhaps by casting a distinctive ballot, creating a point of reference with respect to future ballots. If you enter the voting booth immediately after the adversary, she knows that your ballot will contain the segments immediately following her ballot's segments. Therefore, if the adversary can determine the fingerprint of her ballot after the election, she can guess the fingerprint on your ballot. Numerous similar correlation attacks are possible.

[3]Spool-to-spool DRE-VVPAT systems roll a continuous strip of paper from one spool to another. Election observers may record the order in which voters cast their ballots, enabling association of these voters with the ordered ballots.

DRE-VVPAT with Paper Sheets. Using a DRE-VVPAT with standard, disconnected paper sheets does not mitigate all threats of a paper spool. Anyone that can fingerprint these sheets and has some knowledge of the order in which they will be loaded can mount attacks similar to those involving paper spools. In this case, an adversary would need the ability to scan some or all paper ballots prior to the election, to gain some knowledge of the order that these paper sheets will be loaded into the DRE, to observe the order that some or all voters enter the voting booth, and to fingerprint some or all of the resulting paper ballots later.

Paper-Based Voting. The issues with paper-based voting are similar to DRE-VVPAT with paper sheets. In this case, a voter could have the ability to choose his own blank paper ballot (at least in theory—a poll worker may simply hand the voter a ballot in practice), but the voter may be paid, coerced, or confused into making a nonrandom choice, such as taking the top ballot on the pile. Instead we recommend giving each voter an opportunity to re-randomize the pile of ballots, for example, by cutting the deck. Following this shuffling, an adversary would have greater difficulty inferring a relationship between future voters and their ballot fingerprints. To undermine a paper-based voting system, an adversary would need the ability to scan some or all paper ballots prior to the election, to gain some knowledge of the order of these paper sheets at the poll workers' table, to observe the order that some or all voters receive paper ballots (and, if voters can randomly choose or shuffle the ballots, to draw meaningful inferences in spite of this uncertainty), and to fingerprint some or all of the resulting paper ballots later.

Optical Scan Voting. In many ways, optical scan voting machines present a similar scenario to paper-based voting, but these machines also contain a scanner that

may be capable of fingerprinting the ballots that they scan.[4] The optical scan machine may store the fingerprint and possibly votes on each ballot as part of its normal operations and publicly reveal this data later (Section 3.3 describes how this may be helpful for auditing). In this case, a malicious party would need the ability to scan some or all paper ballots prior to the election, to gain some knowledge of the order of these paper sheets, to observe the order that some or all voters receive paper ballots (and, if voters can randomly choose or shuffle the ballots, to draw meaningful inferences in spite of this uncertainty), and to fingerprint some or all of the resulting paper ballots later (or if the voting machine records the fingerprint-vote combinations electronically, to observe these values).[5]

Pre-Completed Ballots. Fingerprints can also enable dangerous coercion attacks utilizing pre-completed ballots. Imagine that an adversary wishes to coerce voters into choosing a particular candidate. The adversary can distribute pre-completed ballots to targeted individuals, recording the fingerprint of the ballot provided to each voter. If the adversary has access to the ballots after the election, she can use the fingerprints to reidentify the distributed ballots. This would allow the adversary to confirm that the provided ballots are in the ballot box and contain the "correct" votes.

[4]Even if infeasible with present machines, it may become feasible as low-cost scanners increase in resolution and gain additional capabilities.

[5]A reviewer of the workshop-published version of this research notes an additional interesting attack. Assume that each paper ballot contains a stub that includes a serial number, and these stubs are removed for privacy reasons before a voter submits her ballot. Suppose that an adversary can scan ballots and associate fingerprints with each serial number prior to the election. If the adversary can observe the serial number of a ballot given to a voter, that adversary immediately knows the fingerprint of that voter's ballot. This attack emphasizes that, even if removed, serial numbers or other identifiable attributes on a ballot can threaten voter privacy.

Mitigation. The various attack scenarios that we discuss each place certain requirements on an attacker. By making these requirements more difficult to achieve, we may reduce the feasibility of these threats.[6]

In all cases discussed in this section, an adversary must scan some or all paper ballots before voters cast those ballots. To prevent an adversary from producing scans, the paper sheets or rolls should be kept in a locked box whenever possible, and access to the paper should be monitored prior to election day. Election officials should make every effort to prevent the introduction of outside, rather than official, ballots into the ballot box. Scanners and computing devices should be kept away from the paper ballots.

Except in the case of DRE-VVPAT with paper spools, an adversary needs the ability to completely or partially learn the order of these paper sheets. This can be mitigated by shuffling the sheets immediately prior to election day, assuming that an adversary is unable to fingerprint the ballots between the shuffling and the election. Depending on how well-shuffled the ballots are, the order information necessary to mount an attack may be destroyed. Shuffling is far from ideal, but it can make attacks harder.

These scenarios also rely on an adversary's ability to observe the order that voters obtain their paper ballots or the order that these voters enter the voting booth. Given the need for poll workers and observers to view the process, we unfortunately cannot eliminate these possibilities. To mitigate the threats, however, voters should have the ability (if reasonably possible) to shuffle or otherwise re-randomize the pile of paper ballots. This protects other voters too, as randomization of ballots increases an adversary's uncertainly in the relationship between voters and fingerprints.

[6]We do not consider mitigation techniques that seek to make paper ballots more difficult to fingerprint by modifying the ballots or using a different material. Even if possible, such solutions are likely both to be costly and to be vulnerable to future specialized attacks.

In many cases, an adversary requires the ability to scan certain ballots following the election. Except as necessary for auditing and other processes, scanners should not be allowed near the ballots following the election. In general, used ballots should be stored securely in a monitored location.

As described earlier, some optical scan machines may record fingerprint-vote combinations electronically, and the adversary may use this information to reidentify voters' ballots. These values should be stored and revealed only to the degree necessary to conduct the election and audit processes.

No countermeasure discussed in this section is perfect on its own. In addition, some countermeasures may inhibit other necessary goals, such as the ability to conduct a secure, efficient audit. Election officials may choose to focus their efforts on a limited number of feasible countermeasures to eliminate this threat. A cautious ballot shuffling process probably has the greatest potential to eliminate the threat of fingerprinting to ballot secrecy with minimal impact on the remainder of the election process.

3.2.2 Individual Ballot Fingerprinting

In this scenario, a voter reveals his votes to a third party, perhaps under coercion or to permit vote selling. Suppose that a coercer tells a voter to scan his paper ballot between receiving and submitting it and to return a fingerprint of the scan to the coercer. After the election, the coercer can verify fingerprints to identify the voter's ballot among the set of legitimate ballots. If the coercer does not find the ballot or if the ballot contains "incorrect" votes, the voter may face repercussions.

Similarly, a voter may sell her vote by providing a purchaser (rather than a coercer) with a scan or fingerprint of her ballot. If a legitimate ballot exists with that fingerprint and contains the correct votes, the purchaser can pay the voter.

Mitigation. Attacks based on the ability of individual voters to fingerprint their ballots are difficult to prevent. If voters have the covert ability to measure inherent, unique physical properties of their own ballots, election officials are left with few methods of recourse. Our best option to mitigate this threat is disallowing scanners or similar devices near the ballots throughout the election process. While the prospect of a voter sneaking a scanner into the voting booth may seem far-fetched, handheld scanners, increasing-quality cell phone cameras, and other technological innovations are increasing the practicality of these attacks.

Arguably, voters have long had the ability to make their ballots stand out through unique write-in choices or combinations of votes. The threat of fingerprinting differs, however, because a fingerprinted ballot is reidentifiable even if no affirmative steps are taken to make the ballot unique.

3.3 Auditing Techniques

The popularity of paper ballots is partially a consequence of concerns regarding flaws and vulnerabilities in computerized voting systems. Software cannot change a paper ballot already in a ballot box, making voter-verified paper ballots a popular mitigation strategy. These paper ballots are only useful if someone verifies that they are consistent with the electronically tabulated outcome.

Rather than recount all paper ballots to verify the election results, we may examine some subset of these ballots to draw statistical inferences about the election's outcome. The most popular, widely used auditing approach is precinct-based auditing (e.g., [6, 8, 7, 88, 93]). With precinct-based auditing, officials and other parties randomly select some subset of election precincts. For the selected precincts, officials manually count the votes on all paper ballots and ensure that they match the electronic results.

Sampling at a finer level of granularity than precincts can allow for equally strong statistical inferences with fewer paper ballots manually reviewed. The finest level of granularity possible is ballot-based auditing, in which auditors select some subset of electronic ballots and ensure that they match their corresponding paper ballots (e.g., [18, 78, 53]). Ballot-based auditing presents a number of subtle challenges. For example, it can be difficult to ensure that ballots are selected at random without compromising ballot secrecy.

Calandrino et al. describe a machine-assisted auditing method that allows ballot-based auditing yet strives to preserve ballot secrecy [18]. Following the election, an auditing machine prints serial numbers on paper ballots and rescans the ballots, storing the serial numbers and votes electronically. If these votes sum to the initially reported electronic tally, auditors randomly select electronic ballots based on serial number and manually verify that they match the corresponding paper ballots. The scheme in [18] is able to detect discrepancies even if the auditing machine misbehaves. We propose a method that allows ballot-based auditing without printing serial numbers on ballots, instead relying on paper ballot fingerprints.

For our auditing method, we make several assumptions. First, we assume that precincts maintain a sign-in list that observers may monitor as voters enter and leave a polling place. As is standard practice, the sign-in list is made public after the election. This list allows anyone to determine an accurate count of the number of voters. We assume that every voter signing in casts a ballot. In practice, voters rarely sign in without casting a ballot, and we leave methods for ascertaining the number of ballots cast to future work (any issues with obtaining an accurate count affects all known auditing schemes, not just the ones in this chapter). Our auditing scheme checks for discrepancies not only between the paper and electronic records but between those records and the totals from the sign-in list.

Many ballot-based auditing schemes assume that the set of paper ballots contains no added ballots. Section 3.3.2 describes serious additional threats that are possible if an adversary is able to add paper ballots to the ballot box—even if the set of correct paper ballots remains. These attacks can be difficult to detect, traditionally requiring an accurate count of the paper ballots. We discuss this threat and describe how to use fingerprints to detect added paper ballots more efficiently.

Auditors might also want the ability to verify that the set of paper ballots in the ballot box matches the ones delivered to the polling place before the election. This capability enables a number of possibilities. For example, officials could find the set of legitimate paper ballots in the event of ballot-box stuffing. Section 3.3.3 describes how to perform this check by fingerprinting ballots immediately prior to the election.

3.3.1 Fingerprint-Based Auditing

We first discuss a scheme for using fingerprints to detect mismatches from the electronic ballots to the corresponding paper ballots in the ballot box. Our process is designed such that, if at least B incorrect electronic ballots exist, we will find one or more discrepancies with probability greater than or equal to a desired confidence level c.[7] This scheme is primarily for demonstrative purposes, as it requires revelation of the fingerprint and full combination of votes for each ballot cast, potentially posing serious privacy concerns. This requirement undermines ballot secrecy if a voter can distinguish her ballot through her votes. Later, we discuss how to combine fingerprint-based auditing with other techniques to remove this problematic disclosure while preserving the benefits of our approach.

[7] In practice, we want the audit process to determine whether we can be confident in the election's outcome even if we observe a small number of discrepancies. Although we use the more conventional goal of election auditing in this chapter, our methods can extend to meet the more ambitious practical goal.

Procedure. Throughout this section, we assume use of an optical scan voting machine. Our scheme works as follows. When a voter submits her paper ballot, the voting machine records both the ballot's fingerprint and a vector of the votes contained on that ballot. At the end of the election, officials immediately publish both these fingerprint-vote pairs and the voter sign-in sheet (as discussed earlier, the sign-in sheet should be public throughout election day). Therefore, any citizen will have the ability to confirm that the number of electronic ballots matches the number of signed-in voters and that the votes posted add up to the reported tallies. Our final step is to sample from the electronic ballots and ensure that matching paper ballots exist. Otherwise, a discrepancy exists. Note that switching a vote on a single ballot from Candidate A to Candidate B affects the margin between the candidates by two: Candidate A loses one vote and Candidate B gains one vote. Therefore, we seek to reject the hypothesis that $B = \mathit{margin}/2$ incorrect electronic ballots exist.

To sample from the electronic ballots, we make a list of the (precinct, fingerprint, vote vector) triples, one for each ballot, ordered lexicographically. We will sample items from this list and ensure that a ballot containing the proper fingerprint and vote vector exists in the given precinct for each item selected. Two possible sampling methods exist, which we adapt from [18, 8, 7]. The first option is to sample a fixed number of items from this list. We call this the fixed sample size method. Given N total reported ballots, a minimum of $B = \mathit{margin}/2$ incorrect electronic ballots, and a desired confidence level of c, we require a minimum sample size, n, of:

$$n = \min \left\{ u \mid 1 - \prod_{k=0}^{u-1} \frac{N - B - k}{N - k} \geq c \right\} \tag{3.1}$$

Alternatively, we may select each electronic ballot independently with probability p, where $p \geq 1 - (1 - c)^{1/B}$. This latter approach yields a variable sample size and

results in marginally more ballots selected on average, but it is more amenable to optimizations, as discussed below. We call this the variable sample size method.

Given n or p, a number of existing papers describe how to securely and efficiently sample from a list of items, and we refer the reader to those papers for greater detail (see [17, 25, 18]).

When a ballot is sampled, auditors feed ballots from that precinct into a scanner. The scanner stops when it observes a match for that ballot's electronically reported fingerprint. All auditors and observers may verify that the vote vectors match, and observers may use their own scanners to verify the fingerprint.[8] Because this check only verifies that a paper ballot exists matching each sampled electronic ballot, observers in this process only need the ability to personally scan the paper ballots that reportedly match the selected electronic ones—not all paper ballots—making the process far more efficient.

Table 3.1 compares the number of ballots manually reviewed with these methods to the number manually reviewed with precinct-based auditing (using the methods in [93]) for races with margins under 20% in the 2006 Virginia general election. For example, in Virginia's 2006 Webb-Allen U.S. Senate race with a margin of 0.39%, the fixed sample size fingerprinting method requires manual review of 2,337 of 2,370,445 ballots, and the varying sample size method requires review of 2,339 ballots (on average). Precinct-based auditing requires manual review of 1,141,900 ballots (on average).

Ballot-based techniques may be feasible in scenarios for which precinct-based auditing would be impractical. As auditing occurs at a finer degree of granularity, manual review of fewer ballots is necessary to achieve equivalent confidence in a contest's

[8]This process and several others in this chapter require some trust in a computer system—generally, a participant's scanner. This requirement is partially mitigated by the fact that participants may use their own devices (as opposed to devices provided by the jurisdiction) and that fingerprinting requires only relatively inexpensive, multipurpose components available from a variety of manufacturers and retailers.

General Election					
Race	Totals		Fixed Sample Size	Varying Sample Size	Precinct-Based Auditing
	# Votes	Margin	# Ballots (Manual)	# Ballots (Manual)	# Ballots (Manual)
U.S. Senate	2,370,445	0.39%	2,337	2,339	1,141,900
Const. Amnd.	2,328,224	14.12%	63	65	8,062
U.S. House	173,159	2.82%	325	327	62,469
U.S. House	212,079	19.19%	46	48	1,958
U.S. House	241,134	16.36%	54	56	6,120
U.S. House	235,280	11.88%	76	77	12,991
Delegate	14,963	5.75%	157	159	11,442
Average	796,469	8.11%	437	439	177,849

Table 3.1: Ballot-based auditing with fingerprints (for both fixed and varying sample size methods) vs. precinct-based auditing on 2006 Virginia general election data. The first column is the race and the second is the total votes cast and margin separating the top two candidates. The third, fourth, and fifth columns are the number of ballots that would be manually reviewed under our fixed sample size approach, our varying sample size approach, and a precinct-based auditing approach. For example, 2,370,445 votes were cast in the 2006 Virginia U.S. Senate race, resulting in a 0.39% margin separating the top two finishers. Our fixed and varying sample size approaches would require manual review of 2,337 and 2,339 ballots respectively, and a precinct-based auditing method would require manual review of 1,141,900 ballots.

outcome [6]. For jurisdictions with large precincts, ballot-based auditing provides a far more efficient alternative to precinct-based auditing.[9]

Additional techniques are possible to reduce the number of ballots to be sampled, potentially resulting in dramatic efficiency gains. For example, we may take the reported contents of ballots into account when determining the probability that we review each of those ballots. Given that all vote vectors must be public for this auditing method, techniques that consider ballot contents are straightforward to apply. See [18] for details.

This auditing process requires that fingerprint-vote vector combinations be released and that scanners be allowed near the ballots following the election. As discussed earlier in this section and in Section 3.2, these choices may enable certain attack scenarios, so officials must carefully utilize other countermeasures to ensure ballot secrecy. In addition, the practical efficiency and simplicity of this process are unclear and require additional testing. As an alternative, one could check the electronic ballot to paper ballot correspondence using machine-assisted auditing [18], reducing the number of ballots for which the vote combination is revealed publicly (with that technique, only precincts containing sampled ballots must reveal vote vectors). The techniques in the following sections would remain applicable.

3.3.2 A Paper-to-Electronic Check

Suppose that in addition to an ability to change electronic records, an adversary has the ability to add paper ballots to the ballot box. This may be true for a number of reasons. For example, a compromised DRE may print additional paper ballots, or an official may push extra ballots through the slot of a locked ballot box. The

[9] In the 2008 general election, New Hampshire had multiple polling places with more than 10,000 ballots cast. The state delegates decisions regarding the size and location of polling places to individual towns (private communication with Anthony Stevens, Assistant Secretary of State of New Hampshire, April 2012). Ballot-based auditing methods can provide efficiency even if towns decide to tally and store ballots in large batches.

previous check verifies that each electronic ballot has a corresponding paper ballot. If an adversary can add paper ballots, however, the "corresponding paper ballots" might be fraudulently added ballots while the legitimate ballots may be excluded from the electronic results. This would allow an adversary to steal all electronic ballots in a precinct, yet every electronic ballot would have a matching paper ballot (the ballot box would just have many extra non-matching ballots). In this case, we must check not only that each electronic ballot has a corresponding paper ballot but also that each paper ballot has a corresponding electronic ballot. Traditionally, this would require a count of the number of paper ballots in all precincts, but fingerprinting can allow more efficient approaches.

Note that we ignore an adversary that also has the ability to remove paper ballots from the ballot box. Election procedures often dictate that the ballot box should only be unlocked under the watchful eyes of observers. Further, an adversary that can also remove ballots could commit fraud that would be undetectable from the paper and electronic records alone. Such an adversary could arbitrarily modify the paper ballots to match any electronic results.

We now discuss a scheme for using fingerprints to detect whether paper ballots exist without corresponding electronic ballots. This process is designed such that, if at least B paper ballots have no corresponding electronic ballots, we will find at least one with probability greater than or equal to confidence level c.[10] This method could be combined with any electronic-to-paper check, including the one in the previous section or machine-assisted auditing.

As in the previous section, we assume use of an optical scan voting machine. When a voter submits her paper ballot, the voting machine records the ballot's fingerprint (but not the vote vector, decreasing the risk to voter privacy). At the end of election day, officials publish both the fingerprints and the voter sign-in sheet. Any

[10]Like in the previous section, these methods can be extended to account for a small number of discrepancies.

interested party can confirm that the reported number of ballots matches the number of voters. As the final step, election officials and other interested parties will have the opportunity to select paper ballots from precincts' ballot boxes and ensure that they match the published fingerprints from those precincts. The key property is that we can achieve the desired level of confidence in the election's outcome as long as any participant selects randomly from the ballot box.

A note on combined fraud methods. Suppose an adversary can choose to modify electronic ballots without adding paper ballots (forcing us to check that electronic ballots have corresponding paper ballots, as discussed in the previous section) or to modify electronic ballots while adding matching paper ballots (forcing us to check that paper ballots have corresponding electronic ballots, as discussed in this section). That adversary may simultaneously take both approaches. Recall from the previous section that changing a single electronic ballot alone can affect the margin by two. Under an equivalent argument, changing a single electronic ballot and adding a matching paper one can affect the margin by two. Therefore, regardless of approach taken, an adversary must commit $margin/2$ total "instances" of fraud to change the election's outcome. By the pigeonhole principle, the adversary must commit at least $margin/4$ instances of at least one form of fraud. Therefore, to provide confidence c, it is sufficient to apply each auditing approach (paper-to-electronic and electronic-to-paper) to check for $margin/4$ errors with confidence c.

Procedure. In each precinct, we allow each participant in the audit process to select m paper ballots from the ballot box, fingerprint those ballots under observation of other participants, and replace the ballots before the next participant repeats the process. If m is larger than the precinct size (m will normally be small, making this an atypical case), participants may either fingerprint the precinct's reported number of ballots or ignore fingerprints and simply count that the number of ballots is no more

than the reported number of ballots. Participants, which may include representatives of losing candidates, have an incentive to ensure that they select ballots at random from the pile.[11]

We select m such that no matter which precincts additional ballots are in, if more than B total added ballots exist (see note above on choosing B), we will find one with probability at least c. Assuming that v_i ballots were reported in precinct i, an adversary cannot benefit from adding more than v_i fraudulent paper ballots to that precinct. Adding v_i ballots allows the adversary to create an ideally chosen fraudulent ballot in place of each legitimate ballot without exceeding the reported ballot totals. This fact bounds the fraud committed by an optimal adversary in each precinct.

Since auditors sample an equal number of ballots from each precinct, an adversary's optimal strategy is to add fraudulent ballots in the smallest number of precincts. Therefore, let $v_1, v_2, \ldots, v_t$ be the reported number of ballots cast in each precinct in decreasing order. Let B_i represent the number of additional ballots an optimal adversary would add to precinct i. Starting with the largest precinct, we subtract the reported number of ballots cast in each precinct from B until the remaining amount is equal to or smaller than the next precinct, precinct r. B_r equals this remainder, and $B_i = v_i$ for $i < r$. Given these values, we set m such that:

$$m = \min \left\{ u \mid 1 - \prod_{k=0}^{u-1} \left(\prod_{j=1}^{r} \frac{v_j - k}{v_j + B_j - k} \right) \geq c \right\} \tag{3.2}$$

m is guaranteed to be less than or equal to the size of v_r. To see this, note that the adversary must add fraudulent ballots to at least one precinct of size v_r or smaller to achieve the desired level of fraud. If $m \geq v_i$, an auditor must remove at least v_i ballots from precinct i's ballot box, so a non-empty ballot box would be immediate evidence of fraudulently added ballots. If $m = v_r$, an auditor must notice additional

[11] If we are concerned that participants may be unable to select ballots entirely at random, we may increase m to provide equal confidence.

ballots in at least one precinct for which the adversary must add ballots, guaranteeing detection of the fraud. Recall that if $m \geq v_i$, auditors may forgo fingerprinting and simply count that precinct i has no more than v_i paper ballots

When participants perform this sampling, if they see a ballot not on the reported list of fingerprints for the precinct, this indicates that additional ballots are in the ballot box. The security of this scheme rests on an adversary's inability to produce another ballot with the same fingerprint as a legitimate ballot. These methods rely on difficult-to-duplicate properties of paper and are not possible with serial numbers. If we sample a ballot with serial number 10 from a ballot box, no guarantee exists that another ballot with the same serial number is not in the same box. If we sample a ballot with a given fingerprint from a ballot box, we may reasonably believe that no additional ballots with the same fingerprint are in the same box, even if an adversary has attempted to undermine this property.

Note that we may sample ballots for two races simultaneously. If we need to select x ballots from a precinct for race 1 and y ballots for race 2, we may simply select $\max(x, y)$ ballots from that precinct. Also note that, if desired, these methods could be used to sample ballots from the ballot boxes of individual voting machines rather than full precincts.

The method proposed in this section is only one of several approaches that would provide an appropriate level of confidence in the final result. For example, we may instead sample full precincts and count all ballots in those precincts. In this case, auditors would need to review the same number of precincts as is necessary for precinct-based auditing, but auditors would simply need to count the ballots for a paper-to-electronic check (as opposed to checking the votes on those ballots, as is necessary for a traditional precinct-based audit).[12]

[12]One interesting possibility that this raises is that we may use the negative-exponential auditing method of [7] for selecting precincts. Because the probability of selecting a precinct with that method is the same as the likelihood that we check a ballot in a precinct with our electronic-to-paper check (see [18] for details), we may simply recount all ballots in precincts considered for the electronic-

3.3.3 Reconciling Paper Ballots

An additional property that elections officials may wish to check is whether the set of paper ballots in the ballot box is the same as the set of ballots delivered to the precinct on election day. To do so, officials may pre-scan and publish fingerprints for the paper ballots prior to election day. Immediately prior to the election, election officials, candidates, etc. may verify that these published fingerprints are correct by checking that paper ballots exist matching the published fingerprints and that no extra paper ballots exist (using methods like those in the previous two sections). In this way, participants may draw statistically strong conclusions that the set of paper ballots matches the published fingerprints. Note that scanning ballots would make reshuffling or an equivalent procedure necessary prior to the ballots' distribution; otherwise, ordering could permit inference of voters' choices (see Section 3.2).

When voters submit their paper ballots, the optical scan machine may store fingerprints for the ballots.[13] Following the election, officials may sample fingerprints and paper ballots to verify that legitimate ballots ended up in the ballot box and that the set of reportedly unused ballots were actually unused. Among other things, this check would allow auditors to isolate the set of legitimate ballots in a ballot box, helping to deter attacks that rely on slipping in fake ballots.

3.4 Conclusion

Paper fingerprinting poses both challenges and opportunities for election officials. This chapter outlines several threats to ballot secrecy due to recent advances in paper identification and suggests mitigation strategies to counter these threats. While

to-paper check—provided that we selected each ballot with a fixed probability, as discussed in the previous section. This means that we audit each precinct with probability $1 - (1 - p)^{v_i}$ and can ignore precincts skipped by the electronic-to-paper review.

[13]The machine could even reject ballots with invalid fingerprints, though officials should not trust that the machine performs this check.

the most obvious consequences of paper identification are negative, it can also help improve election integrity. Fingerprints can enable an efficient post-election audit process and help detect and prevent additional threats to election integrity.

As technology and algorithms improve, it may be possible for digital cameras and other handheld devices to fingerprint ballots. These new advances will pose additional risks to ballot privacy and should be addressed by future work. For the near future, however, paper will likely remain a critical component of the voting process due to its reliability, cost, familiarity to the public, and ability to stymie many threats to electronic voting systems.

Chapter 4

Bubble Biometrics

Scantron-style fill-in-the-bubble forms are a popular means of obtaining human responses to multiple-choice questions. Whether conducting surveys, academic tests, or—as explored in the previous chapter—elections, these forms allow straightforward user completion and fast, accurate machine input. Although not every use of bubble forms demands anonymity, common perception suggests that bubble completion does not result in distinctive marks. In this chapter, we demonstrate that this assumption is false under certain scenarios, enabling use of these markings as a biometric. The ability to uncover identifying bubble marking patterns has far-reaching potential implications, from detecting cheating on standardized tests to threatening the anonymity of election ballots.

Bubble forms are widely used in scenarios where confirming or protecting the identity of respondents is critical. Over 137 million registered voters in the United States reside in precincts with optical scan voting machines [97], which traditionally use fill-in-the-bubble paper ballots. Voter privacy (and certain forms of fraud) relies on an inability to connect voters with these ballots. Surveys for research and other purposes use bubble forms to automate data collection. The anonymity of survey subjects not only affects subject honesty but also impacts requirements governing

human subjects research [96]. Over 1.6 million members of the high school class of 2010 completed the SAT [24], one of many large-scale standardized tests using bubble sheets. Educators, testing services, and other stakeholders have incentives to detect cheating on these tests. The implications of our findings extend to any use of bubble forms for which the ability to "fingerprint" respondents may have consequences, positive or negative.

Our contributions. We develop techniques to extract distinctive patterns from markings on completed bubble forms. These patterns serve as a biometric for the form respondent. To account for the limited characteristics available from markings, we apply a novel combination of image processing and machine learning techniques to extract features and determine which are distinctive (see Section 4.1). These features can enable discovery of respondents' identities or of connections between completed bubbles.

To evaluate our results on real-world data, we use a corpus of over ninety answer sheets from an unrelated survey of high school students (see Section 4.2). We train on a subset of completed bubbles from each form, effectively extracting a biometric for the corresponding respondent. After training, we obtain a test set of additional bubbles from each form and classify each test set. For certain parameters, our algorithms' top match is correct over 50% percent of the time, and the correct value falls in the top 3 matches 75% percent of the time. In addition, we test our ability to detect when someone other than the expected respondent completes a form, simultaneously achieving false positive and false negative rates below 10%. We conduct limited additional tests to confirm our results and explore details available from bubble markings.

Depending on the application, these techniques can have positive or negative repercussions (see Section 4.3). Analysis of answer sheets for standardized tests

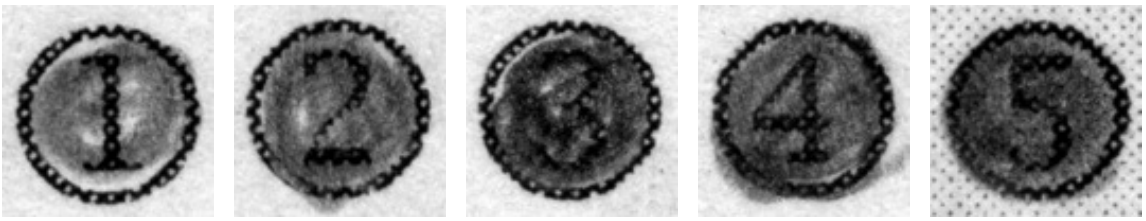
could provide evidence of cheating by test-takers, proctors, or other parties. Similarly, scrutiny of optical-scan ballots could uncover evidence of ballot-box stuffing and other forms of election fraud. With further improvements in accuracy, the methods developed could even enable new forms of authentication. Unfortunately, the techniques could also undermine the secret ballot and anonymous surveys. For example, some jurisdictions publish scanned images of ballots following elections, and employers could match these ballots against bubble-form employment applications. Bubble markings serve as a biometric even on forms and surveys otherwise containing no identifying information. We discuss methods for minimizing the negative impact of this work while exploiting its positive uses (see Section 4.4).

Because our test data is somewhat limited, we discuss the value of future additional tests (see Section 4.6). For example, longitudinal data would allow us to better understand the stability of an individual's distinguishing features over time, and stability is critical for most uses discussed in the previous paragraph.

4.1 Learning Distinctive Features

Filling in a bubble is a narrow, straightforward task. Consequently, the space for inadvertent variation is relatively constrained. The major characteristics of a filled-in bubble are consistent across the image population—most are relatively circular and dark in similar locations with slight imperfections—resulting in a largely homogeneous set. See Figure 4.1. This creates a challenge in capturing the unique qualities of each bubble and extrapolating a respondent's identity from them.

We assume that all respondents start from the same original state—an empty bubble with a number inscribed corresponding to the answer choice (e.g., choices 1-5 in Figure 4.1). When a respondent fills in a bubble, opportunities for variation include the pressure applied to the drawing instrument, the drawing motions employed, and



(a) Person 1 (b) Person 2 (c) Person 3 (d) Person 4 (e) Person 4 - Gray

Figure 4.1: Example marked bubbles. The background color is white in all examples except Figure 4.1(e), which is gray.

the care demonstrated in uniformly darkening the entire bubble. In this work, we consider applications for which it would be infeasible to monitor the exact position, pressure, and velocity of pencil motions throughout the coloring process.[1] In other contexts, such as signature verification, these details can be useful. This information would only strengthen our results and would be helpful to consider if performing bubble-based authentication, as discussed in Section 4.3.

4.1.1 Generating a Bubble Feature Vector

Image recognition techniques often use feature vectors to concisely represent the important characteristics of an image. As applied to bubbles, a feature vector should capture the unique ways that a mark differs from a perfectly completed bubble, focusing on characteristics that tend to distinguish respondents. Because completed bubbles tend to be relatively homogeneous in shape, many common metrics do not work well here. To measure the unique qualities, we generate a feature vector that blends several approaches from the image recognition literature. Specifically, we use PCA, shape descriptors, and a custom bubble color distribution to generate a feature vector for each image.

[1] Clarkson et al. [23] use multiple scans to infer the 3D surface texture of paper, which may suggest details like pressure. We assume multiple scans to be infeasible for our applications.

Principal Component Analysis (PCA) is one common technique for generating a feature set to represent an image [54]. At a high level, PCA reduces the dimensionality of an image, generating a concise set of features that are statistically independent from one another. PCA begins with a sample set of representative images to generate a set of eigenvectors. In most of our experiments, the representative set was comprised of 368 images and contained at least one image for each (respondent, answer choice) pair. Each representative image is normalized and treated as a column in a matrix. PCA extracts a set of eigenvectors from this matrix, forming a basis. We retain the 100 eigenvectors with the highest weight. These eigenvectors account for approximately 90% of the information contained in the representative images.

To generate the PCA segment of our feature vector, a normalized input image (treated as a column vector) is projected onto the basis defined by the 100 strongest eigenvectors. The feature vector is the image's coordinates in this vector space—i.e., the weights on the eigenvectors. Because PCA is such a general technique, it may fail to capture certain context-specific geometric characteristics when working exclusively with marked bubbles.

To compensate for the limitations of PCA, we capture shape details of each bubble using a set of geometric descriptors and capture color variations using a custom metric. Peura et al. [84] describe a diverse a set of geometric descriptors that measure statistics about various shapes. This set includes a shape's center of mass, the center and radius of a circle approximating its shape, and variance of the shape from the approximating circle's radius (see Figure 4.2). The approximating circle minimizes the sum of squared radius deviations. We apply the specified descriptors to capture properties of a marked bubble's boundary. Instead of generating these descriptors for the full marked bubble alone, we also generate the center of mass, mean radius, and radial variance for "sectors" of the marked bubble. To form these sectors, we first evenly divide each dot into twenty-four 15° "slices." Sectors are the 24 overlapping

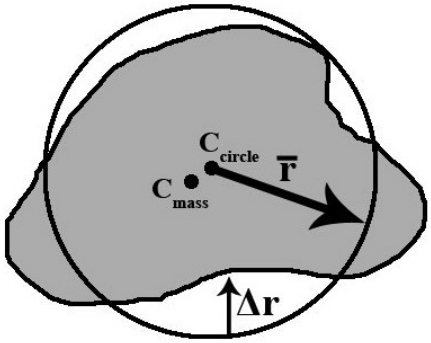


Figure 4.2: An example bubble marking with an approximating circle. The circle minimizes the sum of the squared deviation from the radius. We calculate the circle's center and mean radius, the marking's variance from the radius, and the marking's center of mass.

pairs of adjacent slices (see Figure 4.3). Together, these geometric descriptors add 368 features.

Finally, we developed and use a simple custom metric to represent color details. We divide a dot into sectors as in the previous paragraph. For each sector, we create a histogram of the grayscale values for the sector consisting of fifteen buckets. We throw away the darkest bucket, as these pixels often represent the black ink of the circle border and answer choice numbering. Color distribution therefore adds an additional 14 features for each sector, or a total of 336 additional features.

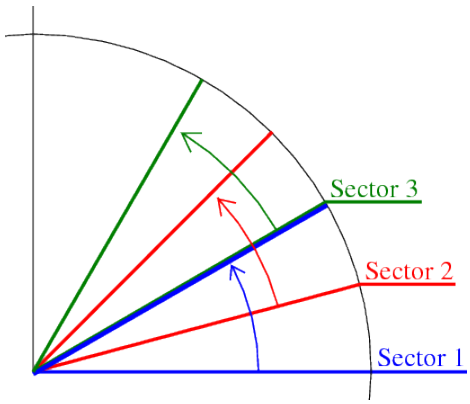


Figure 4.3: Each dot is split into twenty-four 15° slices. Adjacent slices are combined to form a sector, spanning 30°. The first few sectors are depicted here.

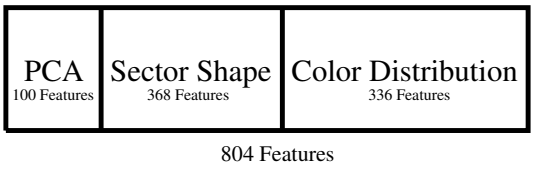


Figure 4.4: Feature vector components and their contributions to the final feature vector length.

The resulting feature vector consists of 804 features that describe shape and color details for a dot and each of its constituent sectors (see Figure 4.4). See Section 4.2.3, where we evaluate the benefits of this combination of features. Given feature vectors, we can apply machine learning techniques to infer distinguishing details and differentiate between individuals.

4.1.2 Identifying Distinguishing Features

Once a set of feature vectors are generated for the relevant dots, we use machine learning to identify and utilize the important features. Our analysis tools make heavy use of WEKA, a popular Java-based machine learning workbench that provides a variety of pre-implemented learning methods [46]. In all experiments, we used WEKA version 3.6.3.

We apply WEKA's implementation of the Sequential Minimal Optimization (SMO) supervised learning algorithm to infer distinctive features of respondents and classify images. SMO is an efficient method for training support vector machines [85]. WEKA can accept a training dataset as input, use the training set and learning algorithm to create a model, and evaluate the model on a test set. In classifying individual data points, WEKA internally generates a distribution over possible classes, choosing the class with the highest weight. For us, this distribution is useful in ranking the respondents believed to be responsible for a dot. We built glue code to collect and process both internal and exposed WEKA data efficiently.

4.2 Evaluation

To evaluate our methods, we obtained a corpus of 154 surveys distributed to high school students for research unrelated to our study. Although each survey is ten pages, the first page contained direct identifying information and was removed prior to our access. Each of the nine available pages contains approximately ten questions, and each question has five possible answers, selected by completing round bubbles numbered 1-5 (as shown in Figure 4.1).

From the corpus of surveys, we removed any completed in pen to avoid training on writing utensil or pen color.[2] Because answer choices are numbered, some risk exists of training on answer choice rather than marking patterns—e.g., respondent X tends to select bubbles with "4" in the background. For readability, survey questions alternate between a white background and a gray background. To avoid training bias, we included only surveys containing at least five choices for each answer 1-4 on a white background (except where stated otherwise), leaving us with 92 surveys.

For the 92 surveys meeting our criteria, we scanned the documents using an Epson v700 Scanner at 1200 DPI. We developed tools to automatically identify, extract, and label marked bubbles by question answered and choice selected. After running these tools on the scanned images, we manually inspected the resulting images to ensure accurate extraction and labeling.

Due to the criteria that we imposed on the surveys, each survey considered has at least twenty marked bubbles on a white background, with five bubbles for the "1" answer, five for the "2" answer, five for the "3" answer, and five for the "4" answer.[3] For each experiment, we selected our training and test sets randomly from this set of twenty bubbles, ensuring that sets have equal numbers of "1," "2," "3," and "4"

[2]We note that respondents failing to use pencil or to complete the survey anecdotally tended not to be cautious about filling in the bubbles completely. Therefore, these respondents may be more distinguishable than those whose surveys were included in our experiment.

[3]To keep a relatively large number of surveys, we did not consider the number of "5" answers and do not use these answers in our analysis.

bubbles for each respondent and trying to balance the number of bubbles for each answer choice when possible.

In all experiments, a random subset of the training set was selected and used to generate eigenvectors for PCA. We required that this subset contain at least one example from each respondent for each of the four relevant answer choices but placed no additional constraints on selection. For each dot in the training and test sets, we generated a feature vector using PCA, geometric descriptors, and color distribution, as described in Section 4.1.1.

We conducted two primary experiments and a number of complementary experiments. The first major test explores our ability to reidentify a respondent from a test set of eight marks given a training set of twelve marks per respondent. The second evaluates our ability to detect when someone other than the official respondent completes a bubble form. To investigate the potential of bubble markings and confirm our results, we conducted seven additional experiments. We repeated each experiment ten times and report the average of these runs.

Recall from Section 4.1.2 that we can rank the respondents based on how strongly we believe each one to be responsible for a dot. For example, the respondent that created a dot could be the first choice or fiftieth choice of our algorithms. A number of our graphs effectively plot a cumulative distribution showing the percent of test cases for which the true corresponding respondent falls at or above a certain rank—e.g., for 75% of respondents in the test set, the respondent's true identity is in the top three guesses.

4.2.1 Respondent Reidentification

This experiment measured the ability to reidentify individuals from their bubble marking patterns. For this test, we trained our model using twelve sample bubbles per respondent, including three bubbles for each answer choice 1-4. Our test set

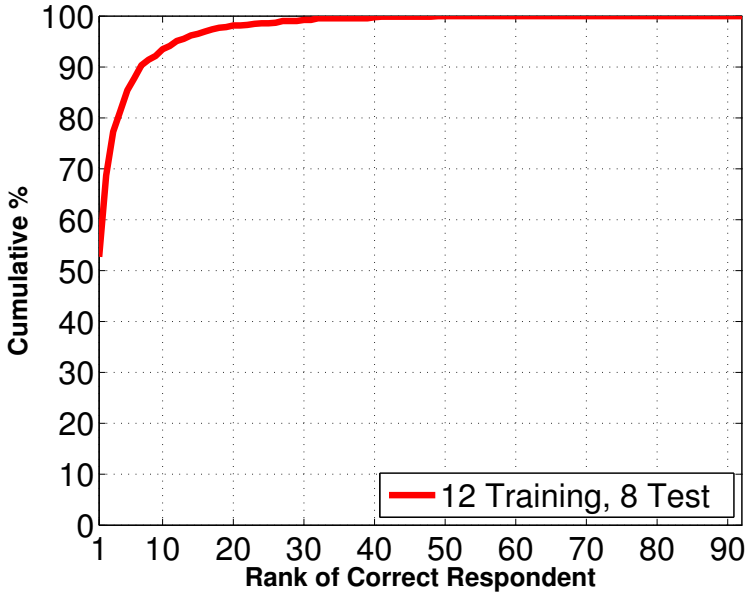


Figure 4.5: Respondent reidentification with 12 training bubbles and 8 test bubbles per respondent.

for each respondent contained the remaining two bubbles for each answer choice, for a total of eight test bubbles. We applied the trained model to each of the 92 respondents' test sets and determined whether the predicted identity was correct.

To use multiple marks per respondent in the test set, we classify the marks individually, yielding a distribution over the respondents for each mark in the set. After obtaining the distribution for each test bubble in a group, we combine this data by averaging the values for each respondent. Our algorithms then order the respondents from highest to lowest average confidence, with highest confidence corresponding to the top choice.

On average, our algorithm's first guess identified the correct respondent with 51.1% accuracy. The correct respondent fell in the top three guesses 75.0% of the time and in the top ten guesses 92.4% of the time. See Figure 4.5, which shows the percentage of test bubbles for which the correct respondent fell at or above each possible rank. This initial result suggests that individuals complete bubbles in a highly distinguishing manner, allowing reidentification with surprisingly high accuracy.

4.2.2 Detecting Unauthorized Respondents

One possible application of this technique is to detect when someone other than the authorized respondent creates a set of bubbles. For example, another person might take a test or survey in place of an authorized respondent. We examined our ability to detect these cases by measuring how often our algorithm would correctly detect a fraudulent respondent who has claimed to be another respondent. We trained our model using twelve training samples from each respondent and examined the output of our model when presented with eight test bubbles. The distribution of these sets is the same as in Section 4.2.1.

For these tests, we set a threshold for the lowest rank accepted as the respondent. For example, suppose that the threshold is 12. To determine whether a given set of test bubbles would be accepted for a given respondent, we apply our trained model to the test set. If the respondent's identity appears in any of the top 12 (of 92) positions in the ranked list of respondents, that test set would be accepted for the respondent. For each respondent, we apply the trained model both to the respondent's own test bubbles and to the 91 other respondents' test bubbles.

We used two metrics to assess the performance of our algorithms in this scenario. The first, false positive rate, measures the probability that a given respondent would be rejected (labeled a cheater) for bubbles that the respondent actually completed. The second metric, false negative rate, measures the probability that bubbles completed by any of the 91 other respondents would be accepted as the true respondent's. We varied the threshold from 1 to 92 for our tests. We expected the relationship between threshold and false negative rate to be roughly linear: increasing the threshold by 1 increases the probability that a respondent randomly falls above the threshold for another respondent's test set by roughly 1/92.[4]

[4]This is not exact because the order of these rankings is not entirely random. After all, we seek to rank a respondent as highly as possible for the respondent's own test set.

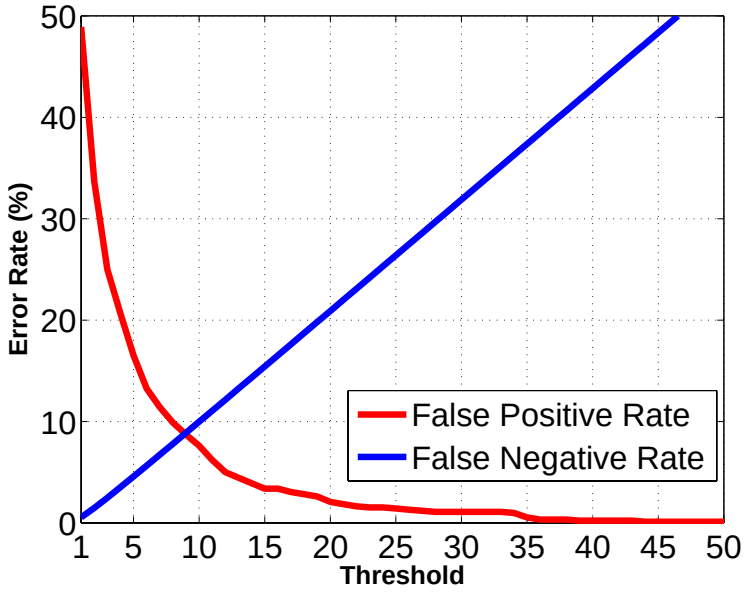


Figure 4.6: False positive and false negative rates when detecting unauthorized respondents.

Our results are presented in Figure 4.6. As we increase the threshold, the false positive rate drops precipitously while the false negative rate increases roughly linearly. If we increase the threshold to 8, then a fraudulent respondent has a 7.8% chance of avoiding detection (by being classified as the true respondent), while the true respondent has a 9.9% chance of being mislabeled a cheater. These error rates intersect with a threshold approximately equal to 9, where the false positive and false negative rates are 8.8%. Assuming a sample that accurately reflects how a larger population completes bubbles and a threshold that increases linearly with sample size, we expect that these results would scale well and eventually stabilize with an increasing number of respondents.

4.2.3 Additional Experiments

To study the information conveyed by bubble markings and support our results, we performed seven complementary experiments. In the first, we evaluate the effect that

scanner resolution has on reidentification accuracy. Next, we considered our ability to reidentify a respondent from a single test mark given a training set containing a single training mark from each respondent. Because bubble forms typically contain multiple markings, this experiment is somewhat artificial, but it hints at the information available from a single dot. The third and fourth supplemental experiments explored the benefits of increasing the training and test set sizes respectively while holding the other set to a single bubble. In the fifth test, we examined the tradeoff between training and test set sizes. The final two experiments validated our results using additional gray bubbles from the sample surveys and demonstrated the benefits of our feature set over PCA alone. As with the primary experiments, we repeated each experiment ten times.

Effect of resolution on accuracy. In practice, high-resolution scans of bubble forms may not be available, but access to lower resolution scans may be feasible. To determine the impact of resolution on reidentification accuracy, we down-sampled each bubble from the original 1200 DPI to 600, 300, 150, and 48 DPI. We then repeated the reidentification experiment of Section 4.2.1 on bubbles at each resolution.

Figure 4.7 shows that decreasing the image resolution has little impact on performance for resolutions above 150 DPI. At 150 DPI, the accuracy of our algorithm's first guess decreases to 45.1% from the 51.1% accuracy observed at 1200 DPI. Accuracy remains relatively strong even at 48 DPI, with the first guess correct 36.4% of the time and the correct respondent falling in the top ten guesses 86.8% of the time. While down-sampling may not perfectly replicate scanning at a lower resolution, these results suggest that strong accuracy remains feasible even at resolutions for which printed text is difficult to read.

Single bubble reidentification. This experiment measured the ability to reidentify an individual using a single marked bubble in the test set and a single example

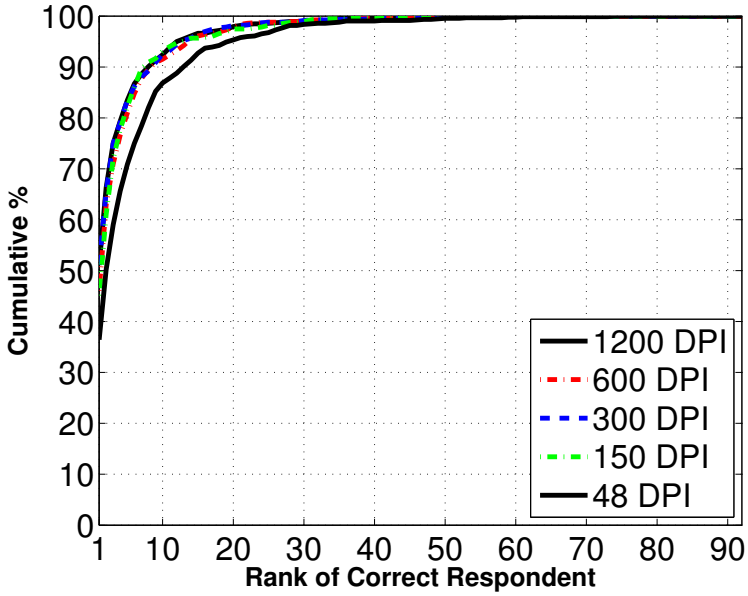


Figure 4.7: Respondent reidentification accuracy using lower-resolution images. Note that the 1200, 600, 300, and 150 DPI lines almost entirely overlap.

per respondent in the training set. This is a worst-case scenario, as bubble forms typically contain multiple markings. We extracted two bubbles from each survey and trained a model using the first bubble.[5] We then applied the trained model to each of the 92 second bubbles and determined whether the predicted identity was correct. Under these constrained circumstances, an accuracy rate above that of random guessing (approximately 1%) would suggest that marked bubbles embed distinguishing features.

On average, our algorithm's first guess identified the correct respondent with 5.3% accuracy, five times better than the expected value for random guessing. See Figure 4.8, which shows the percentage of test bubbles for which the correct respondent fell at or above each possible rank. The correct respondent was in the top ten guesses 31.4% of the time. This result suggests that individuals can inadvertently convey information about their identities from even a single completed bubble.

[5]Note: In this experiment, we removed the restriction that the set of images used to generate eigenvectors for PCA contains an example from each column.

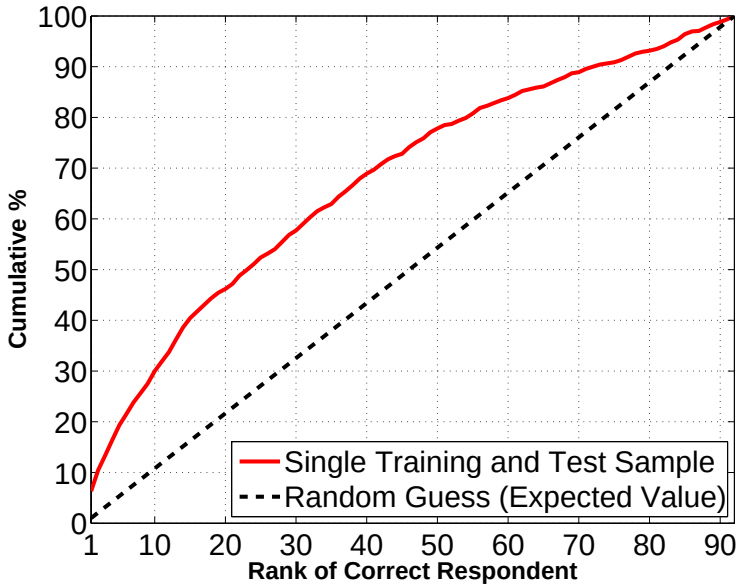


Figure 4.8: One marked bubble per respondent in each of the training and test sets. The expected value from random guessing is provided as reference.

Increasing training set size. In practice, respondents rarely fill out a single bubble on a form, and no two marked bubbles will be exactly the same. By training on multiple bubbles, we can isolate patterns that are consistent and distinguishing for a respondent from ones that are largely random. This experiment sought to verify this intuition by confirming that an increase in the number of training samples per respondent increases accuracy. We held our test set at a single bubble for each respondent and varied the training set size from 1 to 19 bubbles per respondent (recall that we have twenty total bubbles per respondent).

Figure 4.9 shows the impact various training set sizes had on whether the correct respondent was the top guess or fell in the top 3, 5, or 10 guesses. Given nineteen training dots and a single test dot, our first guess was correct 21.8% of the time. The graph demonstrates that a greater number of training examples tends to result in more accurate predictions, even with a single-dot test set. For the nineteen training dots case, the correct respondent was in the top 3 guesses 40.8% of the time and the top 10 guesses 64.5% of the time.

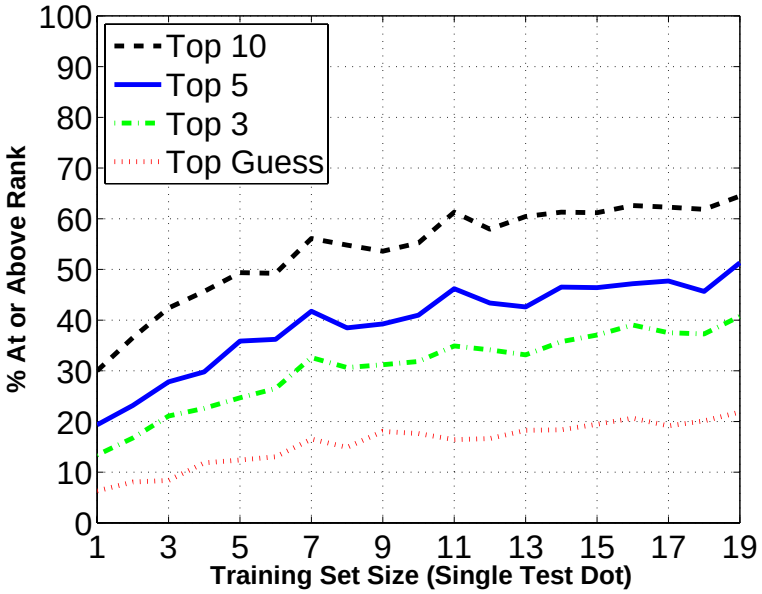


Figure 4.9: Increasing the training set size from 1 to 19 dots per respondent.

Increasing test set size. This experiment is similar to the previous experiment, but we instead held the training set at a single bubble per respondent and varied the test set size from 1 to 19 bubbles per respondent (averaging distributions over test bubbles, as in Section 4.2.1). Intuitively, increasing the number of examples per respondent in the test set helps ensure that our algorithms guess based on consistent features—even if the training set is a single noisy bubble.

Figure 4.10 shows the impact of various test set sizes on whether the correct respondent was the top guess or fell in the top 3, 5, or 10 guesses. We see more gradual improvements when increasing the test set size than observed when increasing training set size in the previous test. From one to nineteen test bubbles per respondent, the accuracy of our top 3 and 5 guesses increases relatively linearly with test set size, yielding maximum improvements of 4.3% and 7.6% respectively. For the top-guess case, accuracy increases with test set size from 5.3% at one bubble per respondent to 8.1% at eight bubbles then roughly plateaus. Similarly, the top 10 guesses case plateaus near ten bubbles and has a maximum improvement of 8.0%. Starting from

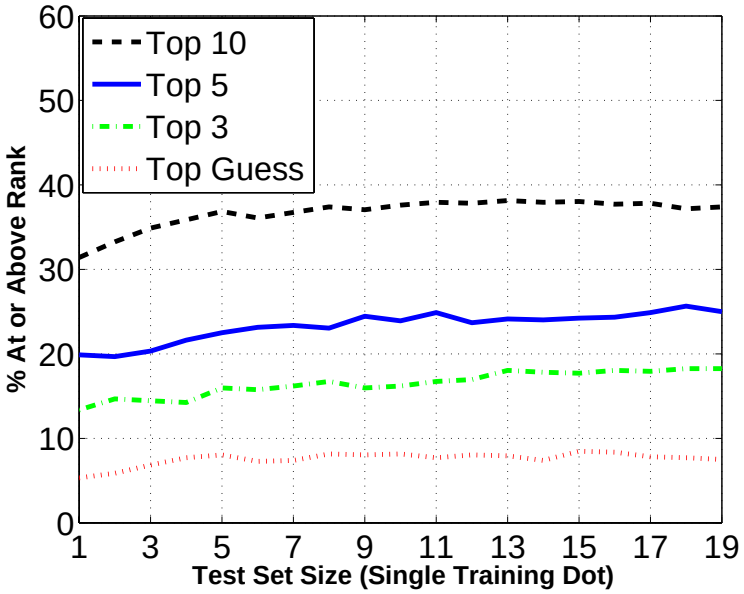


Figure 4.10: Increasing the test set size from 1 to 19 dots per respondent.

equivalent sizes, the marginal returns from increasing the training set size generally exceed those seen as test set size increases. Next, we explore the tradeoff between both set sizes given a fixed total of twenty bubbles per respondent.

Training-test set size tradeoff. Because we have a constraint of twenty bubbles per sample respondent, the combined total size of our training and test sets per respondent is limited to twenty. This experiment examined the tradeoff between the sizes of these sets. For each value of x from 1 to 19, we set the size of the training set per respondent to x and the test set size to $20 - x$. In some scenarios, a person analyzing bubbles would have far larger training and test sets than in this experiment. Fortunately, having more bubbles would not harm performance: an analyst could always choose a subsample of the bubbles if it did. Therefore, our results provide a lower bound for these scenarios.

Figure 4.11 shows how varying training/test set sizes affected whether the correct respondent was the top guess or fell in the top 3, 5, or 10 guesses. As the graph

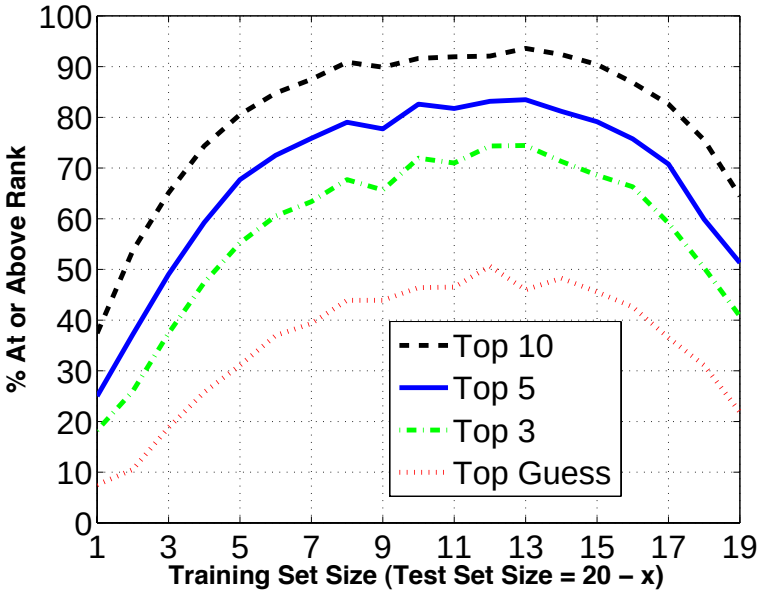


Figure 4.11: Trade-off between training and test set sizes.

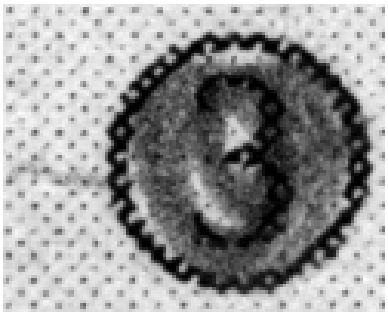


Figure 4.12: This respondent tends to have a circular pattern with a flourish stroke at the end. The gray background makes the flourish stroke harder to detect.

demonstrates, the optimal tradeoff was achieved with roughly twelve bubbles per respondent in the training set and eight bubbles per respondent in the test set.

Validation with gray bubbles. To further validate our methods, we tested the accuracy of our algorithms with a set of bubbles that we previously excluded: bubbles with gray backgrounds. These bubbles pose a significant challenge as the paper has both a grayish hue and a regular pattern of darker spots. This not only makes it harder to distinguish between gray pencil lead and the paper background but also limits differences in color distribution between users. See Figure 4.12.

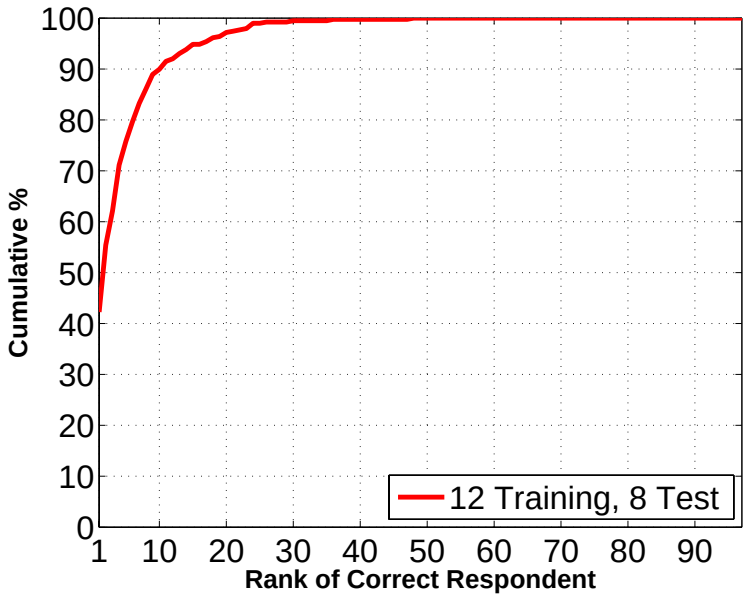


Figure 4.13: Using the unmodified algorithm with the same configuration as in Figure 4.5 on dots with gray backgrounds, we see only a mild decrease in accuracy.

As before, we selected surveys by locating ones with five completed (gray) bubbles for each answer choice, 1-4, yielding 97 surveys. We use twelve bubbles per respondent in the training set and eight bubbles in the test set, and we apply the same algorithms and parameters for this test as the test in Section 4.2.1 on a white background.

Figure 4.13 shows the percentage of test cases for which the correct respondent fell at or above each possible rank. Our first guess is correct 42.3% of the time, with the correct respondent falling in the top 3, 5, and 10 guesses 62.1%, 75.8%, and 90.0% of the time respectively. While slightly weaker than the results on a white background for reasons specified above, this experiment suggests that our strong results are not simply a byproduct of our initial dataset.

Feature vector options. As discussed in Section 4.1.1, our feature vectors combine PCA data, shape descriptors, and a custom color distribution to compensate for the limited data available from bubble markings. We tested the performance of our

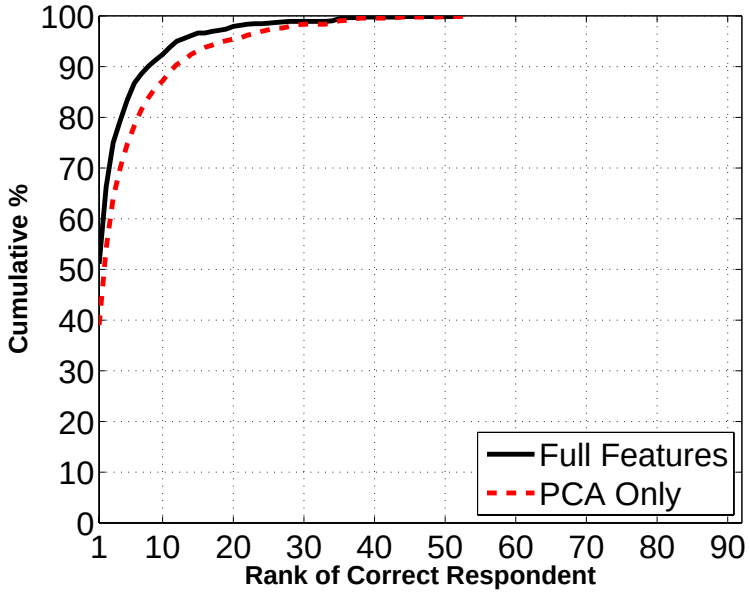


Figure 4.14: Performance with various combinations of features.

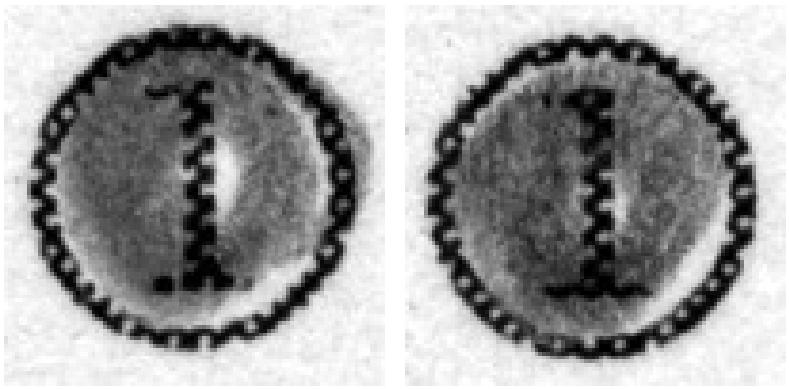
algorithms for equivalent parameters with PCA alone and with all three features combined. This test ran under the same setup as Figure 4.5 in Section 4.2.1.

For both PCA and the full feature set, Figure 4.14 shows the percentage of test cases for which the correct respondent fell at or above each possible rank. The additional features improve the accuracy of our algorithm's first guess from 39.0% to 51.1% and the accuracy of the top ten guesses from 87.2% to 92.4%.

4.2.4 Discussion

Although our accuracy exceeds 50% for respondent reidentification, the restrictive nature of marking a bubble limits the distinguishability between users. We briefly consider a challenging case here.

Figure 4.15 shows marked bubbles from two respondents that our algorithm often mistakes for one another. Both individuals appear to use similar techniques to complete a bubble: a circular motion that seldom deviates from the circle boundary, leaving white-space both in the center and at similar locations near the border.



(a) Person A (b) Person B

Figure 4.15: Bubbles from respondents often mistaken for each other. Both respondents use superficially similar techniques, leaving unmarked space in similar locations.

Unless the minor differences between these bubbles are consistently demonstrated by the corresponding respondents, differentiating between these cases could prove quite difficult. The task of completing a bubble is constrained enough that close cases are nearly inevitable. In spite of these challenges, however, reidentification and detection of unauthorized respondents are feasible in practice.

4.3 Impact

This work has both positive and negative implications depending on the context and application. While we limit our discussion to standardized tests, elections, surveys, and authentication, the ability to derive distinctive bubble completion patterns for individuals may have consequences beyond those examined here. In Section 4.6, we discuss additional tests that would allow us to better assess the impact in several of these scenarios. In particular, most of these cases assume that an individual's distinguishing features remain relatively stable over time, and tests on longitudinal data are necessary to evaluate this assumption.

4.3.1 Standardized Tests

Scores on standardized tests may affect academic progress, job prospects, educator advancement, and school funding, among other possibilities. These high stakes provide an incentive for numerous parties to cheat and for numerous other parties to ensure the validity of the results. In certain cheating scenarios, another party answers questions on behalf of an official test-taker. For example, a surrogate could perform the entire test, or a proctor could change answer sheets after the test [44, 60]. The ability to detect when someone other than the authorized test-taker completes some or all of the answers on a bubble form could help deter this form of cheating.

Depending on the specific threat and available data, several uses of our techniques exist. Given past answer sheets, test registration forms, or other bubbles ostensibly from the same test-takers, we could train a model as in Section 4.2.2 and use it to infer whether a surrogate completed some or all of a test.[6] Although the surrogate may not be in the training set, we may rely on the fact that the surrogate is less likely to have bubble patterns similar to the authorized test-taker than to another set of test-takers. Because our techniques are automated, they could flag the most anomalous cases—i.e., the cases that would be rejected even under the least restrictive thresholds—in large-scale datasets for manual review.

If concern exists that someone changed certain answers after the test (for example, a proctor corrected the first five answers for all tests), we could search for questions that are correctly answered at an usually high rate.[7] Given this information, multiple possible analysis techniques exists. For example, we could train on the less suspicious questions and use the techniques of Section 4.2.2 to determine whether the suspicious ones on a form are from the same test-taker. Alternatively, we could train on the non-

[6]Note our assumption that the same unauthorized individual has not completed both the training bubbles and the current answer sheet.

[7]Assuming that someone may have corrected answers, we could treat all correct answers as suspicious and trust incorrect ones as coming from the original test-taker.

suspicious answer choices from each form and the suspicious answer choices from *all* forms other than a form of interest. Given this model, we could apply the techniques of Section 4.2.1 to see whether suspicious bubbles on that form more closely match less-suspicious bubbles on the same form or suspicious bubbles on other forms.

4.3.2 Elections

Our techniques provide a powerful tool for detecting certain forms of election fraud but also pose a threat to voter privacy.

Suppose that concerns exist that certain paper ballots were fraudulently submitted by someone other than a valid voter. Although the identity of voters might not be known for direct comparison to past ballots or other forms, we can compare batches of ballots in one election to batches from past elections. Accounting for demographic changes and the fact that voters need not vote in all elections, the ballots should be somewhat similar across elections. For example, if 85% of the voters on the previous election's sign-in list also voted during this election, we would expect similarities between 85% of the old ballots and the new ballots. More detailed statistical analysis is left for future work.

Similarly, analysis could also help uncover fraudulent absentee ballots. Because absentee ballots do not require a voter to be physically present, concerns exist about individuals fraudulently obtaining and submitting these ballots [68]. By training a model on past ballots, we could assess whether suspicious absentee ballots fail to match the diversity expected from the population completing these forms.[8]

Unfortunately, because bubble markings can serve as a biometric, they can also be used in combination with seemingly innocuous auxiliary data to undermine ballot secrecy. Some jurisdictions now release scanned images of ballots following elections with the goal of increasing transparency (e.g., Humboldt County, California [49],

[8]If a state uses bubble form absentee ballot applications, analysis could even occur on the applications themselves.

which releases ballot scans at 300 DPI). If someone has access to these images or otherwise has the ability to obtain ballot scans, they can attempt to undermine voter privacy. Although elections may be decided by millions of voters, an attacker could focus exclusively on ballots cast in a target's precinct. New Jersey readjusts larger election districts to contain fewer than 750 registered voters [80]. Assuming 50% turnout, ballots in these districts would fall in groups of 375 or smaller. In Wisconsin's contentious 2011 State Supreme Court race, 71% of reported votes cast fell in the 91% of Wisconsin wards with 1,000 or fewer total votes [99].

Suppose that an interested party, such as a potential employer, wishes to determine how you voted. Given the ability to obtain bubble markings known to be from you (for example, on an employment application), that party can replicate our experiment in Section 4.2.1 to isolate one or a small subset of potential corresponding ballots. What makes this breach of privacy troubling is that it occurs without the consent of the voter and requires no special access to the ballots (unlike paper fingerprinting techniques [16], which require access to the physical ballot). The voter has not attempted to make an identifying mark, but the act of voting results in identifying marks nonetheless. This threat exists not only in traditional government elections but also in union and other elections.

Finally, one known threat against voting systems is pattern voting. For this threat, an attacker coerces a voter to select a preferred option in a relevant race and an unusual combination of choices for the other races. The unusual voting pattern will allow the attacker to locate the ballot later and confirm that the voter selected the correct choice for the relevant race. One proposed solution for pattern voting is to cut ballots apart to separate votes in individual contests [20]. Our work raises the possibility that physically divided portions of a ballot could be connected, undermining this mitigation strategy.[9]

[9]We thank an anonymous reviewer of the symposium-published version of our work for suggesting this possibility.

4.3.3 Surveys

Human subjects research is governed by a variety of restrictions and best practices intended to properly weigh research interests against the subjects' interests. One factor to be considered when collecting certain forms of data is the level of anonymity afforded to subjects. If a dataset contains identifying information, such as subject name, this may impact the types of data that should be collected and procedural safeguards imposed to protect subject privacy. If subjects provide data using bubble forms, these markings effectively serve as a form of identifying information, tying the form to the subject even in the absence of a name. Reidentification of subjects can proceed in the same manner as reidentification of voters, by matching marks from a known individual against completed surveys (as in Section 4.2.1).

Regardless of whether ethical or legal questions are raised by the ability to identify survey respondents, this ability might affect the honesty of respondents who are aware of the issue. Dishonesty poses a problem even for commercial surveys that do not adhere to the typical practices of human subjects research.

The impact of this work for surveys is not entirely negative, however. In certain scenarios, the person responsible for administering a survey may complete the forms herself or modify completed forms, whether to avoid the work of conducting the survey or to yield a desired outcome. Should this risk exist, similar analysis to the standardized test and election cases could help uncover the issue.

4.3.4 Authentication

Because bubble markings are a biometric, they may be used alone or in combination with other techniques for authentication. Using a finger or a stylus, an individual could fill in a bubble on a pad or a touchscreen. Because a computer could monitor user input, various details such as velocity and pressure could also be collected and used to increase the accuracy of identification, potentially achieving far stronger re-

sults than in Section 4.2.2. On touchscreen devices, this technique may or may not be easier for users than entry of numeric codes or passwords. Additional testing would be necessary for this application, including tests of its performance in the presence of persistent adversaries.

4.4 Mitigation

The impact of our techniques can be both beneficial and detrimental, but the drawbacks may outweigh the benefits under certain circumstances. In these cases, a mitigation strategy is desirable, but the appropriate strategy varies. We discuss three classes of mitigation strategies. First, we consider changes to the forms themselves or how individuals mark the forms. Second, we examine procedural safeguards that restrict access to forms or scanned images. Finally, we explore techniques that obscure or remove identifying characteristics from scanned images. No strategy alone is perfect, but various combinations may be acceptable under different circumstances.

4.4.1 Changes to Forms or Marking Devices

As explored in Section 4.2.3, changes to the forms themselves such as a gray background can impact the accuracy of our tests. The unintentional protection provided by this particular change was mild and unlikely to be insurmountable. Nevertheless, more dramatic changes to either the forms themselves or the ways people mark them could provide a greater measure of defense.

Changes to the forms themselves should strive to limit either the space for observable human variation or the ability of an analyst to perceive these variations. The addition of a random speckled or striped background in the same color as the writing instrument could create difficulties in cleanly identifying and matching a mark. If bubbles had wider borders, respondents would be less likely to color outside the

lines, decreasing this source of information. Bubbles of different shapes or alternate marking techniques could encourage less variation between users. For example, some optical scan ballots require a voter simply to draw a line to complete an arrow shape [5], and these lines may provide less identifying information than a completed bubble.

The marking instruments that individuals use could also help leak less identifying information. Some Los Angeles County voters use ink-marking devices, which stamp a circle of ink for a user [65]. Use of an ink-stamper would reduce the distinguishability of markings, and even a wide marker could reduce the space for inadvertent variation.

4.4.2 Procedural Safeguards

Procedural safeguards that restrict access to both forms themselves and scanned images can be both straightforward and effective. Collection of data from bubble forms typically relies on scanning the forms, but a scanner need not retain image data for any longer than required to process a respondent's choices. If the form and its image are unavailable to an adversary, our techniques would be infeasible.

In some cases, instructive practices or alternative techniques already exist. For example, researchers conducting surveys could treat forms with bubble markings in the same manner as they would treat other forms containing identifying information. In the context of elections, some jurisdictions currently release scanned ballot images following an election to provides a measure of transparency. This release is not a satisfactory replacement for a statistically significant manual audit of paper ballots (e.g., [8, 18]), however, and it is not necessary for such an audit. Because scanned images could be manipulated or replaced, statistically significant manual confirmation of the reported ballots' validity remains necessary. Furthermore, releasing the recorded choices from a ballot (e.g., Washington selected for President, Lincoln selected for Senator, etc.) without a scanned ballot image is sufficient for a manual audit.

Whether the perceived transparency provided by the release of ballot scans justifies the resulting privacy risk is outside the scope of this work. Nevertheless, should the release of scanned images be desirable, the next section describes methods that strive to protect privacy in the event of a release.

4.4.3 Scrubbing Scanned Images

In some cases, the release of scanned bubble forms themselves might be desirable. In California, Humboldt County's release of ballot image scans following the 2008 election uncovered evidence of a software glitch causing certain ballots to be ignored [47]. Although a manual audit could have caught this error with high probability, ballot images provide some protection against unintentional errors in the absence of such audits.

The ability to remove identifying information from scanned forms while retaining some evidence of a respondent's actions is desirable. One straightforward approach is to cover the respondent's recorded choices with solid black circles. Barring any stray marks or misreadings, this choice would completely remove all identifying bubble patterns. Unfortunately, this approach has several disadvantages. First, a circle could cover choices that were not selected, hiding certain forms of errors. Second, suppose that a bubble is marked but not recorded. While the resulting image would allow reviewers to uncover the error, such marks retain a respondent's identifying details. The threat of a misreading and reidentification could be sufficient to undermine respondent confidence, enabling coercion.

An alternative to the use of black circles is to replace the contents of each bubble with its average color, whether the respondent is or is not believed to have selected the bubble. The rest of the scan could be scrubbed of stray marks. This would reduce the space for variation to color and pressure properties alone. Unfortunately, no evidence exists that these properties cannot still be distinguishing. In addition, an average

might remove a respondent's intent, even when that intent may have been clear to the scanner interpreting the form. Similar mitigation techniques involve blurring the image, reducing the image resolution, or making the image strictly black and white, all of which have similar disadvantages to averaging colors.

One interesting approach comes from the facial image recognition community. Newton et al. [81] describe a method for generating k-anonymous facial images. This technique replaces each face with a "distorted" image that is k-anonymous with respect to faces in the input set. The resulting k-anonymous image maintains the expected visual appearance, that of a human face. The exact details are beyond the scope of this work, but the underlying technique reduces the dimensionality using Principal Component Analysis and an algorithm for removing the most distinctive features of each face [81].

Application of facial anonymization to bubbles is straightforward. Taking marked and unmarked bubbles from all ballots in a set, we can apply the techniques of Newton et al. to each bubble, replacing it with its k-anonymous counterpart. The result would roughly maintain the visual appearance of each bubble while removing certain unique attributes. Unfortunately, this approach is imperfect in this scenario. Replacement of an image might hide a respondent's otherwise obvious intent. In addition, distinguishing trends might occur over multiple bubbles on a form: for example, an individual might mark bubbles differently near the end of forms (this is also a problem for averaging the bubble colors). Finally, concerns exist over the guarantees provided by k-anonymity [14], but the work may be extensible to achieve other notions of privacy, such as differential privacy [34].

We caution that the value of these images for proving the true contents of physical bubble forms is limited: an adversary with access to the images, whether scrubbed or not, could intentionally modify them to match a desired result. These approaches are most useful where the primary concern is unintentional error.

4.5 Related Work

Biometrics. Biometrics can be based on physical or behavioral characteristics of an individual. Physical biometrics are based on physical characteristics of a person, such as fingerprints, facial features, and iris patterns. Behavioral biometrics are based on behavioral characteristics that tend to be stable and difficult to replicate, such as speech or handwriting/signature [52]. Bubble completion patterns are a form of behavioral biometric.

As a biometric, bubble completion patterns are similar to handwriting, though handwriting tends to rely on a richer, less constrained set of available features. In either case, analysis may occur on-line or off-line [74]. In an on-line process, the verifying party may monitor characteristics like stroke speed and pressure. In an off-line process, a verifying party receives only the resulting data, such as a completed bubble. Handwriting-based recognition sometimes occurs in an on-line setting. Because off-line recognition is more generally applicable, our analysis occurred purely in an off-line manner. In some settings, such as authentication, on-line recognition would be possible and could yield stronger results.

Document reidentification. Some work seeks to reidentify a precise physical document for forgery and counterfeiting detection (e.g., [23]). While the presence of biometrics may assist in reidentification, the problems discussed in this chapter differ. We seek to discover whether sets of marked bubbles were produced by the same individual. Our work is agnostic towards whether the sets come from the same form, different forms, or duplicates of forms. Nevertheless, our work and document reidentification provide complementary techniques. For example, document reidentification could help determine whether the bubble form (ballot, answer sheet, survey, etc.) provided to an individual matches the one returned or detect the presence of fraudu-

lently added forms. See the previous chapter for more information in the context of elections.

Cheating Detection. Existing work uses patterns in answer choices themselves as evidence of cheating. Myagkov et al. [73] uncover indicators of election fraud using aggregate vote tallies, turnout, and historical data. Similarly, analysis of answers on standardized tests can be particularly useful in uncovering cheating [43, 60]. For example, if students in a class demonstrate mediocre overall performance on a test yet all correctly answer a series of difficult questions, this may raise concerns of cheating. The general strategy in this line of research is to look for answers that are suspicious in the context of either other answers or auxiliary data.

Bubble-based analysis is also complementary to these anti-cheating measures. Each technique effectively isolates sets of suspicious forms, and the combination of the two would likely be more accurate than each independently. Although our techniques alone do not exploit contextual data, they have the advantage of being unbiased by that data. If a student dramatically improves her study habits, the resulting improvement in test scores alone might be flagged by other anti-cheating measures but not our techniques.

4.6 Future Work

Although a variety of avenues for future work exist, we focus primarily on possibilities for additional testing and application-specific uses here.

Our sample surveys allowed a diverse set of tests, but access to different datasets would enable additional useful tests. We are particularly interested in obtaining and using longitudinal studies—in which a common set of respondents fill out bubble forms multiple times over some period—to evaluate our methods. In addition to providing an increased number of examples, this could also identify how a respondent's markings

vary over time, establish consistency over longer durations, and confirm that our results are not significantly impacted by writing utensil. Because bubble forms from longitudinal studies are not widely available, this might entail collecting the data ourselves.

While we tested our techniques using circular bubbles with numbers inscribed, numerous other form styles exist. In some cases, respondents instead fill in ovals or rectangles. In other cases, selection differs dramatically from the traditional fill-in-the-shape approach—for example, the line-drawing approach discussed in Section 4.4 bears little similarity to our sample forms. Testing these cases would not only explore the limits of our work but could also help uncover mitigation strategies.

Section 4.3 discusses a number of applications of our techniques. Adapting the techniques to work in these scenarios is not always trivial. For example, Section 4.5 discusses existing anti-cheating techniques for standardized tests. Combining the evidence provided by existing techniques and ours would strengthen anti-cheating measures, but it would also require some care to process the data quickly and merge results.

Use of bubble markings for authentication would require both additional testing and additional refinement of our techniques. Given a dataset containing on-line information, such as writing instrument position, velocity, and pressure, we could add this data to our feature vectors and test the accuracy of our techniques with these new features. This additional information could increase identifiability considerably—signature verification is commonly done on-line due to the utility of this data—and may yield an effective authentication system. Depending on the application, a bubble-based authentication system would potentially need to work with a finger rather than a pen or stylus. Because the task of filling in a bubble is relatively constrained, this application would require cautious testing to ensure that an adversary cannot impersonate a legitimate user. This application makes more sense as an out-of-band

authentication technique than as one accompanying data entry: on-line data entry typically involves selecting radio buttons and checkboxes as opposed to scribbling inside bubbles.

4.7 Conclusion

Marking a bubble is an extremely narrow task, but as this work illustrates, the task provides sufficient expressive power for individuals to unintentionally distinguish themselves. Using a dataset with 92 individuals, we demonstrate how to reidentify a respondent's survey with over 50% accuracy. In addition, we are able to detect an unauthorized respondent with over 92% accuracy with a false positive rate below 10%. We achieve these results while performing off-line analysis exclusively, but on-line analysis has the potential to achieve even higher rates of accuracy.

The implications of this study extend to any system utilizing bubble forms to obtain user input, especially cases for which protection or confirmation of a respondent's identity is important. Additional tests can better establish the threat (or benefit) posed in real-world scenarios. Mitigating the amount of information conveyed through marked bubbles is an open problem, and solutions are dependent on the application. For privacy-critical applications, such the publication of ballots, we suggest that groups releasing data consider means of masking respondents' markings prior to publication.

Chapter 5

Privacy and Collaborative Filtering

Recommender systems are ubiquitous on the Web. When you buy products from Amazon, rent movies on Netflix, listen to music on Last.fm, or perform myriad other tasks online, recommender systems make suggestions based on your behavior. They typically rely on *collaborative filtering*, or patterns learned from other users: for example, "customers who buy item X (as you just did) often buy item Y."

We investigate the privacy risks of recommender systems based on collaborative filtering. By design, such systems do not directly reveal behavior of individual users or any "personally identifiable information." Their recommendations are based on aggregated data involving thousands to millions of users, each with dozens to thousands of transactions. Moreover, modern collaborative filtering leverages relationships between items rather than relationships between users, creating an extra level of indirection between public recommendations and individual transactions. One might therefore assume that it is infeasible to draw meaningful inferences about transactions of specific users from the public outputs of recommender systems. We show that this assumption is wrong.

Our contributions. We develop a set of practical algorithms that allow accurate inference of (partial) individual behavior from the aggregate outputs of a typical

recommender system. We focus on item-to-item collaborative filtering, in which the system recommends items similar to a given item. Our key insight is to exploit the dynamics of public recommendations in order to make the leap from aggregate to individual data. This work is the first to make and quantitatively evaluate the observation that temporal changes in aggregate recommendations enable accurate inference of individual inputs.

Our algorithms require only passive, "black-box" access to the public outputs of a recommender system, as available to any Internet user. The attacker need not create fake customers or enter purchases or ratings into the system. We do not assume that customers' transactions are available in either identifiable or anonymized form. Our approach is thus fundamentally different from the techniques for reidentifying anonymized transactional records [76]. Reidentification assumes that the attacker has direct access to customers' records. By contrast, our attacks rely only on indirect access: the records are fed into a complex collaborative filtering algorithm and the attacker's view is limited to the resulting outputs.

Our algorithms monitor changes in the public outputs of recommender systems—item similarity lists or cross-item correlations—over a period of time. This dynamic information is then combined with a moderate amount of auxiliary information about some of the transactions of a particular "target" user. The combination is used to infer many of the target user's unknown transactions with high accuracy. Auxiliary information can be obtained by analyzing the user's publicly revealed behavior; we discuss this in more detail in Section 5.2.

Overview of results. We evaluate our algorithms on real-world recommender systems that produce different types of recommendations. Our goal is not to claim privacy flaws in these specific sites—in fact, we often use data voluntarily disclosed by their users to verify our inferences—but to demonstrate the general feasibility of inferring individual transactions from the outputs of collaborative filtering systems.

Some recommender systems make item-to-item correlations available. An example is Hunch, a popular recommendation and personalization website. There is a tradeoff between the number of inferences and their accuracy. When optimized for accuracy, our algorithm infers a third of the test users' secret answers to Hunch questions with no error.

Other recommender systems make only item similarity or "related items" lists available, with or without numeric similarity scores. Examples include Last.fm, an online music service, and LibraryThing, an online book cataloging service and recommendation engine. The results from our LibraryThing experiment illustrate the yield-accuracy tradeoff, ranging from 58 inferences per user with 50% accuracy to 6 inferences per user with 90% accuracy. Another example of item similarity lists is the "Customers who bought this item also bought ..." feature on Amazon. Our ability to evaluate our algorithms on Amazon's recommender system is constrained by the lack of a "ground-truth oracle" for verifying our inferences, but we conducted a limited experiment to demonstrate the feasibility of adversarial inference against Amazon's recommendations.

By necessity, our experiments on real-world systems involve only a limited sample of users. To demonstrate that our inference algorithms also work at scale, we implemented an item-to-item collaborative filtering engine very similar to that used by Amazon, and ran it on the Netflix Prize dataset of movie-rating histories [79]. This allowed us to simulate a complete system, producing public recommendations as well as auxiliary information about users. The underlying dataset of individual ratings served as the "ground-truth oracle" for verifying inferences made by our algorithm. Our algorithm was able to infer 4.5% of transactions of sufficiently active users with an accuracy of 90%.

There is a passing similarity between our inference algorithms and actual collaborative filtering. Both use statistical methods to reach probabilistic conclusions about

unknown aspects of users' behavior. Our algorithms, however, are technically different and pursue a fundamentally different goal: not to predict future events, but to infer past events. This translates into several concrete differences, discussed in Section 5.4. For example, in contrast to prediction algorithms, ours perform best when a user deviates from normal patterns and if his transactions involve less popular items. We can also infer an approximate date when a transaction occurred.

For completeness with respect to different types of recommender systems, we present a simple active attack on user-based collaborative filtering. In broad terms, the attacker creates multiple sybil users whose transactional profile is similar to what he knows about the target user's profile and infers the target's non-public transactions from the recommendations made by the system to these sybils.

In summary, this work is the first to develop a generic method for inferring information about individual users' transactions from the aggregate outputs of collaborative filtering. We show that public outputs of common recommender algorithms may expose non-public details of individual users' behavior—products they purchase, news stories and books they read, music they listen to, videos they watch, and other choices they make—without their knowledge or consent.

5.1 Survey of Recommender Systems

Recommender systems have become a vital tool for attracting and keeping users on commercial websites. Their utility is supported by research [45] as well as common practice.

The task of a recommender system can be abstractly described as follows. Consider a matrix in which rows correspond to users and columns correspond to items. Each value in this matrix represents a user's revealed or stated preference (if any) for an item: for example, whether he purchased a book, how many times he listened

to a song, or what rating he gave to a movie. Because the item set is typically far larger than a single user can consume and evaluate, this matrix is "sparse:" only a small fraction of entries are filled in. A recommender system takes this matrix as input, along with any available metadata about users (such as demographics) and items (such as item categories). The goal of the system is to extrapolate users' "true" preferences over the full item set.

Recommender systems can provide several types of recommendations. If the system suggests items to an individual user based on its knowledge of the user's behavior, it provides *user-to-item* recommendations. If the system helps users find similar users, it provides *user-to-user* recommendations. If, given an item, the system suggests similar items, it provides *item-to-item* recommendations. The system may even list users that are strongly associated with a given item, thus providing *item-to-user* recommendations. The same system may provide several types of recommendations: for example, Last.fm provides both item-to-item and user-to-user recommendations.

We focus on item-to-item recommendations, both because they are supported by essentially all popular online recommender systems and because their output is typically public and thus the most feasible avenue for an attack.

A thorough technical survey of the literature on recommender systems can be found in [1]. Recommender systems can be classified as content-based, collaborative, and hybrid. Content-based systems identify relationships between items based on metadata alone and recommend items that are similar to the user's past transactions. Purely content-based recommender systems pose no privacy risks under our attacks, since the system does not consider other users' transactions when making recommendations to a user.

Collaborative filtering is much more robust and domain-agnostic, and hence far more popular. Collaborative filtering identifies relationships between items based on the preferences of all users. Traditional collaborative filtering methods are *user-based*.

For a given user, the system finds other users with a similar transaction history. In the user-to-user case, the system recommends these similar users; in the user-to-item case, it recommends items selected by the similar users.

The alternative is *item-based* collaborative filtering, which was first described by Sarwar et al. [91] and has become the dominant approach [63, 64, 11]. It generates recommendations using *item similarity scores* for pairs of items, which are based on the likelihood of the pair being purchased by the same customer. Although some systems make raw similarity scores public, their main uses are internal: for example, to find items that are similar to a user's previously purchased items in order to make user-to-item recommendations.

5.1.1 Item-to-Item Recommendations

It has become standard practice for online recommender systems to publish item-to-item recommendations, usually in the form of item similarity lists produced from item similarity scores. Given an item, these lists help find related items (see [29] for a survey of algorithms). On Amazon, this is seen as the "Customers who bought this item also bought ..." feature. Similar features are found on many commercial websites, including iTunes, Last.fm, Pandora, Netflix, YouTube, Hulu, and Google Reader. Item similarity lists even appear on many sites that do not have traditional user-to-item recommendations, such as IMDb, CNN, and the New York Times.[1] Item similarity lists may be limited to top N items or contain an ordered list of hundreds or thousands of items.

Many systems reveal additional information. Amazon reveals not only the relative popularity of items via bestseller lists and "sales rank," but also the percentage of users purchasing certain other items after viewing the given item.[2] For every song,

[1] Even offline retailers such as supermarkets frequently deploy item-to-item similarity analysis to optimize store layout [13].

[2] We do not exploit the latter information for the inference attacks in this chapter. This is an interesting topic for future research.

Last.fm provides the number of listeners and how many times it was played by each listener. Given a book, LibraryThing provides several ordered lists of related books, including more common and more obscure recommendations; some lists also contain detailed transaction information, such as the precise number of users who have both books. Finally, Hunch gives all users access to the entire item-to-item covariance matrix via an API.

5.1.2 User-to-Item Recommendations

User-to-item recommendations may be user-based (finding similar users and recommending their items) or item-based (finding items related to ones that the user chose in the past). Amazon provides several personalized lists with up to 1,000 items to logged-in users. LibraryThing, Last.fm, and Netflix also provide recommendation lists to their users.

5.2 Attack Model

We view the data that the system uses to make recommendations as a matrix where rows correspond to users and columns to items. Each cell represents a *transaction* (*e.g.*, the user's purchase or stated preference for an item). Entries may be dated; the date may or may not be sensitive from a privacy perspective. As users interact with the system, the matrix is continually updated and new recommendations are generated.

Our primary focus is on passive inference attacks. The attacker has access to the public outputs of the recommender system, which depending on the system, may include item similarity lists, item-to-item covariances, and/or relative popularity of items (see Section 5.1). The outputs available to the attacker are available to any user of the system. Crucially, the attacker observes the system over a certain period and

can thus capture changes in its outputs: an increase in covariance between certain items, appearance of an item on the similarity list of another item, an increase in an item's sales rank, *etc.* Note, however, that each update incorporates the effects of hundreds or thousands of transactions. With the exception of auxiliary information (described below), inputs into our inference algorithms are based on aggregate statistics and contain neither explicitly identifying information nor information about specific transactions.

For completeness, we also briefly consider active attacks, where the attacker creates fake, "sybil" users and manipulates their entries in the corresponding rows of the transaction matrix. Depending on the system, this includes adding new entries (easy in the case of ratings and stated preferences, more expensive for purchases), modifying existing entries, or deleting them (easy in the case of ratings and preferences and may also be possible for purchases; for example, one can instruct Amazon to ignore certain purchases when making recommendations). Observable outputs include items recommended by the system to the sybil users and, in the case of systems like Last.fm or LibraryThing, also user similarity lists, which explicitly identify users with similar histories.

Auxiliary information. We assume that for some users, a subset of their transaction history is available to the attacker. We refer to this as the attacker's auxiliary information. An inference attack is successful if it enables the attacker to learn transactions that are *not* part of the auxiliary information. In other words, the attacker's objective is to "leverage" his prior knowledge of some of the target user's transactions to discover transactions that he did not know about.

There are many sources of auxiliary information. The first is the target system itself. On many websites, users publicly rate or comment on items, revealing a high likelihood of having purchased them. The system may even publicly confirm the purchase, *e.g.*, "verified purchase" on Amazon. Alternatively, on sites with granular

privacy controls, some of the transactions may be publicly visible, while others remain private.

The second source is users revealing partial information about themselves via third-party sites. This is increasingly common: for example, music websites allow embedding of tracks or playlists on blogs or other sites, while Amazon Kindle allows "tweeting" a selected block of text; the identity of the book is automatically shared via the tweet.[3]

The third source is data from other sites that are not directly tied to the user's transactions on the target site, but leak partial information about them. For example, books listed in a Facebook user profile reveal partial information about purchases on Amazon. Linking users across different sites is a well-studied problem [77, 51]. On blippy.com, "a website where people obsessively review everything they buy," individual purchase histories can be looked up by username, making linkages to other sites trivial. Note that the third (and to a lesser extent, the second) source of auxiliary information is outside the control of the recommender system.

Furthermore, information about users' behavior is constantly leaked through public mentions on online fora, real-world interactions with friends, coworkers, and acquaintances, *etc.* Therefore, we do not consider the availability of auxiliary information to be a significant impediment to our attacks.

[3]The stream of such tweets can be conveniently accessed in real time by searching Twitter for "amzn.com/k/".

5.3 Generic Inference Attacks

5.3.1 Inference Attack on Related-Items Lists

In this setting of the problem, the recommender system outputs, for each item, one or more lists of related items. For example, for each book, LibraryThing publishes a list of popular related books and a list of obscure related books.

The description of the inference algorithm in this section is deliberately simplified with many details omitted for clarity. Intuitively, the attacker monitors the similarity list(s) associated with each auxiliary item (*i.e.*, item that he knows to be associated with the target user). The attacker looks for items that either enter the list or move up, indicating increased "similarity" with the auxiliary item. If the same target item t enters and/or moves up in the related-items lists of a sufficiently large subset of the auxiliary items, the attacker infers that t has been added to the target user's record.

Algorithm 1 shows the inference procedure. Intuitively, delta matrices N_Δ store information about the movement of each target item t in the related-items lists of auxiliary items $\mathcal{A}$ (we defer the discussion of matrix construction). The attacker computes a score for each t using a scoring function. The simplest scoring function counts the number of auxiliary items for which t has entered or risen in the corresponding related-items lists. If the final score exceeds a predefined threshold, the attacker concludes that t has been added to the user's record.

Scoring can be significantly more complex, taking into account full dynamics of item movement on related-items lists or giving greater weight to certain lists. To reduce false positives and improve inference accuracy, the scoring function must be fine-tuned for each system. For example, recommender systems tend to naturally cluster items. Netflix users who watched the DVD of the first season of "The Office" also tend to watch the second season. Suppose that some movie rises in the similarity lists of both seasons' DVDs. Because the overlap in viewers across seasons is so great,

Algorithm 1: RELATEDITEMSLISTINFERENCE

Input: Set of target items $\mathcal{T}$, set of auxiliary items $\mathcal{A}$, scoring function : $\mathbb{R}^{|\mathcal{A}|} \to \mathbb{R}$
Output: Subset of items from $\mathcal{T}$ that are believed by the attacker to have been added to the user's record

```
inferredItems = {}
foreach observation time τ do
    Δ = observation period beginning at τ
    N_Δ = delta matrix containing changes in positions of items from T in lists
    associated with items from A
    foreach target item t in N_Δ do
        scores_t = ScoreFunction(N_Δ[t])
        if scores_t ≥ threshold and t ∉ A then
            inferredItems = inferredItems ∪ {t}
return inferredItems
```

this does not reveal much more information than a movie rising in the list associated with a single DVD. In fact, it may reveal *less* if users who watch only one of the two seasons are very unusual.

Our scoring functions prefer sets of auxiliary items that span genres or contain obscure items. Consider LibraryThing, where users share the books they read. Classics such as *To Kill a Mockingbird* or *Catcher in the Rye* are so common that changes in their similarity lists tend to result from widespread trends, not actions of a single user. Movement of a book in a list associated with an obscure book reveals more than movement in a list associated with a bestseller.

5.3.2 Inference Attack on the Covariance Matrix

In this setting of the problem, the item-to-item covariance matrix is visible to any user of the system. An example of an online recommender system that reveals the covariance matrix is Hunch (see Section 5.5.1). We also explain complications, such as asynchronous updates to the system's public outputs, that apply to the related-items scenario as well.

Let $\mathcal{I}$ be the set of items. The recommender system maintains an item-to-item matrix M. For any two distinct items $i, j \in \mathcal{I}$, the (i, j) cell of M contains a measure

of the similarity between i and j. In the setting of this section, (i, j) and (j, i) contain the covariance between i and j. In the setting of Section 5.3.1, the (i, j) cell contains the position, if any, of item i in j's related-items list, along with additional information such as numeric similarity strength. As users interact with the recommender system by making purchases, entering their preferences, *etc.*, the system continually accumulates more data and updates M at discrete intervals.

For each user u, the recommender system maintains a "record" $\mathcal{S}_u \subset \mathcal{I}$. As the user interacts with the system, some item t may be added to $\mathcal{S}_u$, reflecting that t is now related to the user. In some systems, the same item may be added to $\mathcal{S}_u$ multiple times: for example, the user may listen to a particular song, watch a movie, or purchase a product more than once. The system may also remove items from $\mathcal{S}_u$, but this is less common and not used for our attack.

Consider a toy case when a single user u interacts with the system between time τ_1 and $\tau_2 = \tau_1 + \Delta$, and t is added to the user's item list $\mathcal{S}_u$. Covariance between t and all other items in $\mathcal{S}_u$ must increase. Let M_1 be the matrix at time τ_1, M_2 the matrix at time τ_2, and $M_\Delta = M_2 - M_1$. Then, for all items $s_i \in \mathcal{S}_u$, the (s_i, t) entry of M_Δ will be positive. Of course, real-world recommender systems interact concurrently with multiple users whose item sets may overlap.

Intuitively, the attack works as follows. The attacker has auxiliary information about some items in the target user's record (Section 5.2). By observing simultaneous increases in covariances between auxiliary items and some other item t, the attacker can infer that t has been added to the user's record.

Formally, the attacker's auxiliary information is a subset $\mathcal{A} \subseteq \mathcal{S}_u$. It helps—but is not necessary—if $\mathcal{A}$ is uniquely identifying, *i.e.*, for any other user u_j of the recommender system, $\mathcal{A} \nsubseteq \mathcal{S}_{u_j}$. This is possible if items in $\mathcal{A}$ are less popular or if $\mathcal{A}$ is large enough [76].

The attacker monitors the recommender system and obtains the covariance matrix M at each update. Let $\mathcal{T} \subseteq \mathcal{I} \setminus \mathcal{A}$ be the set of items the user may have selected. The attacker observes the submatrix of M formed by rows corresponding to the items in $\mathcal{T} \cup \mathcal{A}$ and columns corresponding to the items in $\mathcal{A}$. Call this submatrix N. Since $\mathcal{A} \subseteq \mathcal{S}_u$, when an item $t \in T$ is added to $\mathcal{S}_u$, covariances between t and many $a_i \in \mathcal{A}$ will increase. If the attacker can accurately recognize this event, he can infer that t has been added to $\mathcal{S}_u$.

The inference procedure is significantly complicated by the fact that when an item is added to $\mathcal{S}_u$, not all of its covariances are updated at the same time due to processing delays. In particular, (t, a_i) covariances for $a_i \in \mathcal{A}$ may update at different times for different auxiliary items a_i. Furthermore, auxiliary items may enter the system at or around the same time as t. We cannot use the (t, a_i) covariance unless we are certain that the addition of item a_i to u's record has been reflected in the system. Before attempting an inference, we compute the subset of auxiliary items that "propagated" into the covariance matrix. The algorithm works by measuring increases in pairwise covariances between auxiliary items. In the following, we refer to this algorithm as PropagatedAux.

Constructing delta matrices. Suppose the attacker observes the covariance submatrices $N_{\tau_1}, N_{\tau_2}, \ldots$ at times $\tau_1, \tau_2, \ldots$. For each observation, the attacker creates a delta matrix N_Δ that captures the relevant *changes* in covariances. There are several ways to build this matrix. In the following, $\tau_{\max}$ is a parameter of the algorithm, representing the upper bound on the length of inference windows.

Strict time interval. For each τ_i, set $N_\Delta = N_{\tau_{i+1}} - N_{\tau_i}$. Since not all covariances may update between τ_i and τ_{i+1}, some entries in N_Δ may be equal to 0.

Algorithm 2: MatrixInference

Input: Set of target items $\mathcal{T}$, set of auxiliary items $\mathcal{A}$, PropagatedAux returns a subset of $\mathcal{A}$, implementation-specific parameters $threshold_{support,score}$

Output: Subset of items from $\mathcal{T}$ that are believed by the attacker to have been added to $\mathcal{S}_u$

$inferredItems = \{\}$
foreach *observation time* τ **do**
  $propagated_\tau$ = PropagatedAux$(\mathcal{A}, \tau)$
  Δ = observation period beginning at τ
  N_Δ = delta matrix containing changes in covariances between items in $\mathcal{T} \cup \mathcal{A}$
  foreach *item* t *in* $\mathcal{T}$ **do**
    $scoreSet_t$ = subset of $a \in \mathcal{A}$ such that $N_\Delta[t][a] > 0$
    $support_t = |scoreSet_t \cap propagated_\tau|$
    $score_t = \frac{|support_t|}{|propagated_\tau|}$
    if $score_t \geq threshold_{score}$ ***and*** $support_t \geq threshold_{support}$ **then**
      $inferredItems = inferredItems \cup \{t\}$
return *inferredItems*

First change. For each τ_i, N_Δ consists of the first changes in covariance after τ_i. Formally, for each entry (x, y) of N, let $\tau_k > \tau_i$ be the first time after τ_i such that $\tau_k \leq \tau_{\max}$ and $N_{\tau_k}[x][y] \neq N_{\tau_i}[x][y]$. Set $N_\Delta[x][y] = N_{\tau_k}[x][y] - N_{\tau_i}[x][y]$.

Largest change. Similar to first change.

Making an inference. The attacker monitors changes in the submatrix N. For each relevant interval Δ, the attacker computes the delta matrix N_Δ as described above and uses PropagatedAux to compute which auxiliary items have propagated into N. Then he applies Algorithm 2. In this algorithm, $scoreSet_t$ is the set of all auxiliary items whose pairwise covariances with t increased, $support_t$ is the subset of $scoreSet_t$ consisting of auxiliary items that have propagated, $score_t$ is the fraction of propagated items whose covariances increased. If $score_t$ and $support_t$ exceed certain thresholds (provided as parameters of the algorithm), the attacker concludes that t has been added to the user's record.

Inference algorithms against real-world recommender systems require fine-tuning and adjustment. Algorithm 2 is only a high-level blueprint; there are many system-

specific variations. For example, the algorithm may look only for increases in covariance that exceed a certain threshold.

5.3.3 Inference Attack on kNN Recommender Systems

Our primary focus is on passive attacks, but for completeness we also describe a simple, yet effective active attack on the *k-nearest neighbor* (kNN) recommendation algorithm [1]. Consider the following user-to-item recommender system. For each user U, it finds the k most similar users according to some similarity metric (*e.g.*, the Pearson correlation coefficient or cosine similarity). Next, it ranks all items purchased or rated by one or more of these k users according to the number of times they have been purchased and recommends them to U in this order. We assume that the recommendation algorithm and its parameters are known to the attacker.

Now consider an attacker whose auxiliary information consists of the user U's partial transaction history, *i.e.*, he already knows m items that U has purchased or rated. His goal is to learn U's transactions that he does not yet know about.

The attacker creates k sybil users and populates each sybil's history with the m items that he knows to be present in the target user U's history. Due to the sparsity of a typical transaction dataset [76], $m \approx O(\log N)$ is sufficient for the attack on an average user, where N is the number of users. (In practice, $m \approx 8$ is sufficient for datasets with hundreds of thousands of users.) With high probability, the k nearest neighbors of each sybil will consist of the other $k-1$ sybils and the target user U. The attacker inspects the list of items recommended by the system to any of the sybils. Any item that appears on the list and is *not* one of the m items from the sybils' artificial history must be an item that U has purchased. Any such item was not previously known to the attacker and learning about it constitutes a privacy breach.

This attack is even more powerful if the attacker can adaptively change the fake history of his sybils after observing the output of the recommender system. This

capability is supported by popular systems—for example, Netflix users can change previously entered ratings, while Amazon users can tell the site to ignore certain transactions when making recommendations—and allows the attacker to target multiple users without having to create new sybils for each one.

5.3.4 Attack Metrics

Our attacks produce inferences of this form: "Item Y was added to the record of user X during time period T." The main metrics are yield and accuracy. Yield is the number of inferences per user per each observation period, regardless of whether those inferences are correct. Accuracy is the percentage of inferences that are correct. We use yield rather than alternative metrics that focus on the number of correct inferences because the attacker can adjust the parameters to control the number of inferences made by our algorithm but cannot directly control the number or proportion that are correct. Where it makes sense, we also express yield as the percentage of the user's transactions inferred by our algorithm, but in general, we focus on the absolute number of inferences.

High yield and high accuracy are not simultaneously necessary for an attack to be dangerous. A single accurate inference could be damaging, revealing anything from a medical condition to political affiliation. Similarly, a large number of less accurate inferences could be problematic if their implications are uniformly negative. While the victim may retain plausible deniability for each individual inference, this provides little or no protection against many privacy violations. For example, plausible deniability does not help in situations where judgments are based on risk (*e.g.*, insurance) or prejudice (*e.g.*, workplace discrimination), or where the inferred information further contributes to a negative narrative (*e.g.*, confirms existing concerns that a spouse is cheating).

There is an inherent tradeoff between yield and accuracy. The higher the yield, the higher the number of incorrect inferences ("false positives"). Different combinations of parameters for our algorithms produce either more inferences at the cost of accuracy, or fewer, but more accurate inferences. Therefore, we evaluate our algorithms using the yield-accuracy curve.

5.4 Inference vs. Prediction

At first glance, our inference algorithms may look similar to standard collaborative filtering algorithms, which attempt to predict the items that a user may like or purchase in the future based on his and other users' past transactions.

The two types of algorithms are dramatically different, both technically and conceptually. We infer the user's actual transactions—as opposed to using the known behavior of similar users to guess what he may do or may have done. Prediction algorithms discover common patterns and thus have low sensitivity to the presence or absence of a single user. Our algorithms are highly sensitive. They (1) work better if there are no similar users in the database, but (2) do not work if the target user is not the database, even if there are many similar users.

Collaborative filtering often exploits covariances between items; our algorithms exploit changes in covariance over time. The accuracy of predictions produced by collaborative filtering does not change dramatically from period to observation period; by contrast, we infer the approximate date when the transaction occurred, which is very hard to discover using collaborative filtering. Finally, our algorithms can infer even transactions involving very obscure items. Such items tend to populate lower ranges of auxiliary items' similarity lists, where a single transaction has the biggest impact. Section 5.6 shows that transactions involving obscure items are more likely to be inferred by our algorithms.

Prediction quality can be seen as a baseline for feasible inference quality. A prediction is effectively an expected probability that a user with item a will select some target item t at any time. If a user with item a selects item t during a given time period, he exceeds this expected probability, causing a temporary rise (until other users balance the impact). By looking at changes in predictions over short periods of time, we can reconstruct how user behavior deviated from the predictions to produce the observed changes. This yields more accurate information than predictions alone. As Sections 5.5.1 and 5.6 show, our algorithms not only outperform a Bayesian predictor operating on the same data, but also infer items ranked poorly by a typical prediction algorithm.

Finally, it is worth mentioning that we use some machine-learning techniques for tuning inference algorithms that operate on related-items lists (see Section 5.5.3). These techniques are very different from collaborative filtering. Whereas collaborative filtering attempts to predict future behavior based on past behavior of other users, our models are backward-facing. We know that an item has risen in a similarity list, but we don't know why. To produce accurate inferences, we must learn which observations are sufficient to conclude that this rise signals addition of the item to the target user's record. In summary, we use machine learning to learn the behavior of the recommender system itself, not the behavior of its users.

5.5 Evaluation on Real-World Systems

We evaluated our inference algorithms on several real-world recommender systems. Our goal was not to carry out an actual attack, but to demonstrate the feasibility and measure the accuracy of our algorithms. Therefore, all experiments were set up so that we knew each user's record in advance because the user either revealed it voluntarily through the system's public interface or cooperated with us. This provided

the "ground-truth oracle," enabling us to measure the accuracy of our inferences without violating anyone's privacy.

5.5.1 Hunch

Hunch.com provides personalized recommendations on a wide range of topics. For each topic, Hunch poses a series of multiple-choice questions to the user and uses the responses to predict the user's preferences. Hunch also has a large set of generic personal questions in the category "Teach Hunch About You" (THAY), intended to improve topic recommendations. Hunch aggregates collected data and publishes statistics that characterize popular opinions in various demographics. For example, according to responses given to Hunch, "birthers" are 94% more likely to say that cultural activities are not important to them and 50% more likely to believe in alien abductions [50].

Statistics collected by Hunch are accessible via an API. They include the number of users responding to each THAY question, the percentage selecting each possible answer, the number of users who responded to each pair of questions, and covariances between each pair of possible answers.

We show that aggregate statistics available via the Hunch API can be used to infer an individual user's responses to THAY questions, even though these responses are not made public by Hunch. Suppose the attacker knows some auxiliary information about a Hunch user (*e.g.*, height, hair color, age, hometown, political views) that allows the attacker to reliably predict how the user will respond to the corresponding THAY questions. We refer to the latter as AUX questions. See Section 5.2 for possible sources of auxiliary information.

Setup. The attacker forms a list of questions consisting of both AUX questions and questions for which he does not know the user's responses. We refer to the latter as TARGET questions; the objective of the experiment is to infer the user's responses

to them. For our experiment, we chose questions with at least four possible answers. There were 375 such questions in the THAY set at the time of our experiment with simulated users (see below), but new THAY questions are continually added and users may even suggest new questions.

Immediately prior to the attack, the attacker uses the API function `responsePairStats` to collect all pairwise covariances between possible answers to questions on his list. Next, he directs the target user to specific questions from his list via links of the form http://hunch.com/people/⟨username⟩/edit-answer/?qid=⟨qid⟩ where ⟨username⟩ is replaced by the target user's username and ⟨qid⟩ is replaced with the question id. The attacker must know the username, but the site provides a social networking feature with profile pages where usernames can be linked to real names, interests, and other personal information. We assume that the user responds to all questions at the same time and that his responses to most AUX questions match the attacker's auxiliary information (our inference algorithms are robust to some mistakes in AUX).

Our goal is to show that individual responses can be inferred from the public outputs of recommender systems, not to conduct an actual attack. Therefore, we omit discussion of mechanisms for convincing a Hunch user to respond to a set of THAY questions. Similarly, it is a matter of opinion which questions and answers constitute sensitive information about an individual. For our purposes, it is sufficient to show that the attacker can infer the values of the user's secret responses to questions chosen by the attacker.

Data collection. Hunch does not update the covariance matrix immediately after the user responds to the attacker-supplied questions. At the time of our experiment, Hunch had approximately 5,000 possible answers to THAY questions and thus had to keep statistics on 12.5 million answer pairs. The update cycle of pairwise statistics varies, but seems to be on the order of two to three weeks. Each day during this

period, for each known AUX response a_i, the attacker uses `responsePairStats` to collect the covariances between (1) a_i and all possible answers to TARGET questions, and (2) a_i and a_j, where $i \neq j$ (*i.e.*, cross-covariances between all AUX responses).

The above covariances are not updated simultaneously, which greatly complicates the attacker's task. Hunch appears to split THAY questions into chunks and update pairwise answer statistics one chunk at a time. For instance, covariances between possible answers to question 1 and question 2 may update on Tuesday and Friday, while covariances between answers to question 1 and question 3 update on Thursday. The attacker must be able to detect when the covariances he is interested in have "propagated" (see Section 5.3.2).

Inferring secret responses. Algorithm 3 shows the inference procedure. Intuitively, the algorithm looks for a subset of AUX answers whose cross-covariances have increased (indicating that they propagated into the covariance matrix), and then for a single answer to each of the TARGET questions whose covariances with most of the AUX responses in the propagated subset have increased simultaneously.

For the algorithm to work, it is essential that large chunks of AUX responses propagate into the covariance matrix at the same time (as is the case for Hunch). The attacker can expect to see large positive shifts in covariance between the user's (known) responses to AUX questions and (unknown) responses to TARGET questions soon after both AUX and TARGET have propagated. The larger the number of AUX questions for which this pattern is observed, the higher the attacker's confidence that the TARGET answer for which covariances have increased is the user's true response.

Results. For the experiment with real users, we used five volunteers and chose THAY questions with at least four possible answers. Questions were ordered by sample size, and each user was assigned twenty questions in a round-robin fashion; fifteen were randomly designated as AUX and five as TARGET. We requested that users respond honestly to all questions and collected their responses to serve as the

Algorithm 3: HUNCHINFERENCE

Input: Set $\mathcal{Q}$ of non-overlapping sets $\mathcal{R}_q$ containing all possible answers to each TARGET question q, set of known responses to AUX questions $\mathcal{A}$, PROPAGATEDAUX returns a subset of $\mathcal{A}$, implementation-specific parameters $threshold_{support,score}$

Output: Inferred responses to each TARGET question

$inferredResponses = \{\}$
foreach *answer set* $\mathcal{R}_q$ *in* $\mathcal{Q}$ **do**
  $maxScore = threshold_{score}$
  $maxSupport = threshold_{support}$
  foreach *observation time* τ **do**
    $propagated_\tau = \text{PROPAGATEDAUX}(\mathcal{A}, \tau)$
    Δ = observation period beginning at τ
    N_Δ = delta matrix containing changes in covariances between items in $\mathcal{R}_q \cup \mathcal{A}$
    foreach *TARGET answer* r *in* $\mathcal{R}_q$ **do**
      $scoreSet_r$ = subset of $a \in \mathcal{A}$ such that $N_\Delta[r][a] > 0$
      $support_r = |scoreSet_r \cap propagated_\tau|$
      $score_r = \frac{|support_r|}{|propagated_\tau|}$
      if $score_r \geq threshold_{score}$ **then**
        if $support_r > maxSupport$ **then**
          $inferredResponses[q] = \{r\}$
          $maxSupport = support_r$
          $maxScore = score_r$
        else if $support_r = maxSupport$ **then**
          if $score_r > maxScore$ **then**
            $maxScore = score_r$
            $inferredResponses[q] = \{r\}$
          else if $score_r == maxScore$ **then**
            $inferredResponses[q] = inferredResponses[q] \cup \{r\}$
return *inferredResponses*

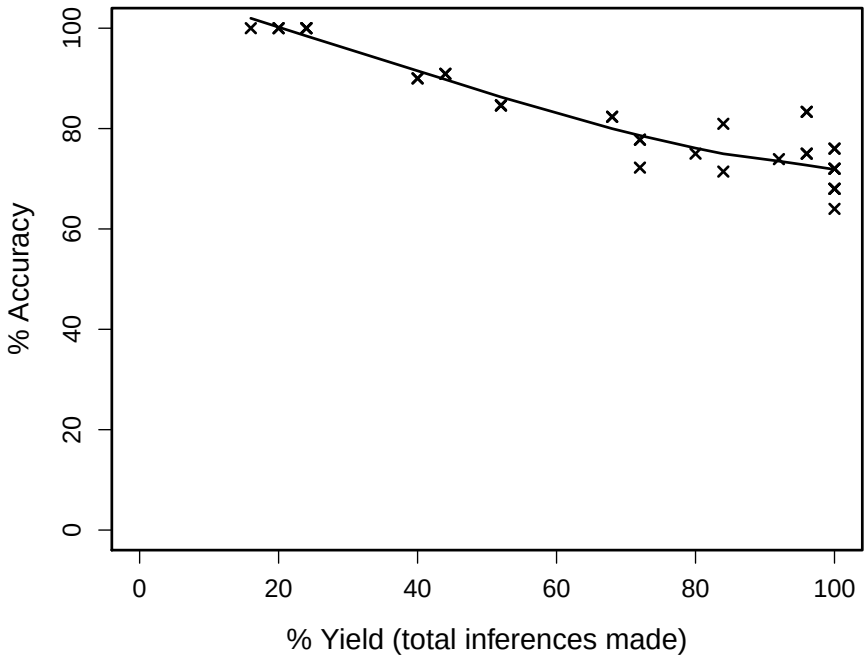


Figure 5.1: Hunch: Accuracy vs. yield for real users. Each point represents a particular tuning of the algorithm, $threshold_{score}$ ranges from 45% to 78%, $threshold_{support}$ ranges between 32% and 57% of AUX size.

"ground-truth oracle." After all responses were entered into Hunch, we collected pairwise answer statistics via the API as described above and applied Algorithm 3 to infer the responses to TARGET questions.

Results are shown in Figure 5.1 in the form of a yield-accuracy curve, with each point corresponding to a particular setting of the algorithm's parameters. We constructed a linear relation between $threshold_{score}$ and $threshold_{support}$ parameters that produced good results across all experiments. We use this relation for all Hunch graphs. Parameter ranges are listed in captions. Here yield is the fraction of unknown responses for which the algorithm produces candidate inferences and accuracy is the fraction of candidate inferences that are correct.

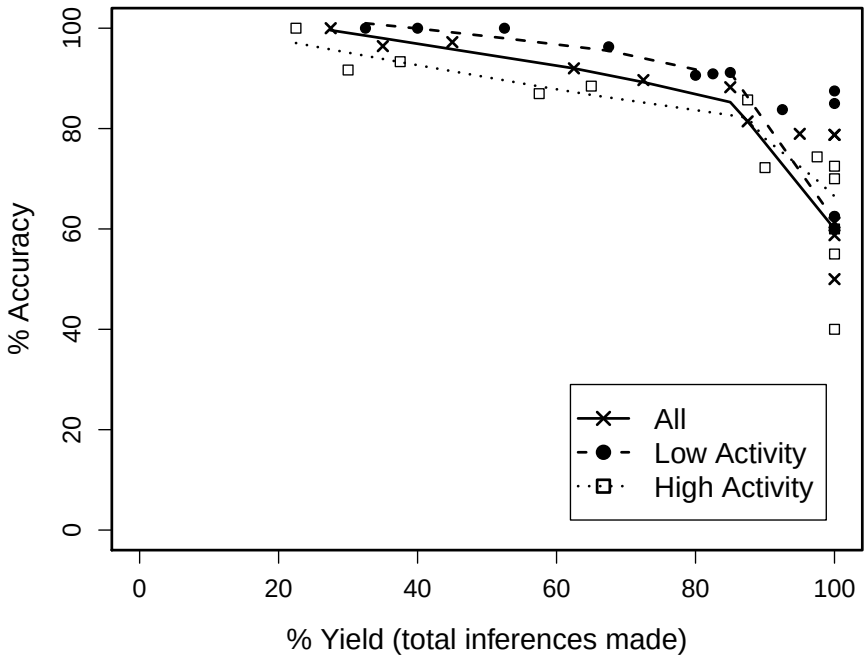


Figure 5.2: Hunch: Accuracy vs. yield for simulated users: average of 8 users, 4 users assigned low-activity questions, 4 users assigned high-activity questions, $threshold_{score}$ ranges from 40% to 75%, $threshold_{support}$ ranges between 28% and 55% of AUX size.

For the experiment on simulated users, we used all 375 Hunch THAY questions with at least 4 possible answers. We monitored the number of users responding to each question (calculated as change in sample size) for one week prior to the experiment and ranked questions by activity level. The forty questions with the lowest activity were assigned to user A, the next forty to user B, *etc.*, for a total of nine users. Due to a data collection error, the data for one user had to be discarded.

For each user, thirty questions were randomly assigned as AUX and ten as TARGET. The simulated users "chose" answers following the actual distribution obtained from Hunch, *e.g.*, if 42% of real users respond "North America" to some question, the simulated user selects this answer with 0.42 probability. Results are in Figure 5.2. As

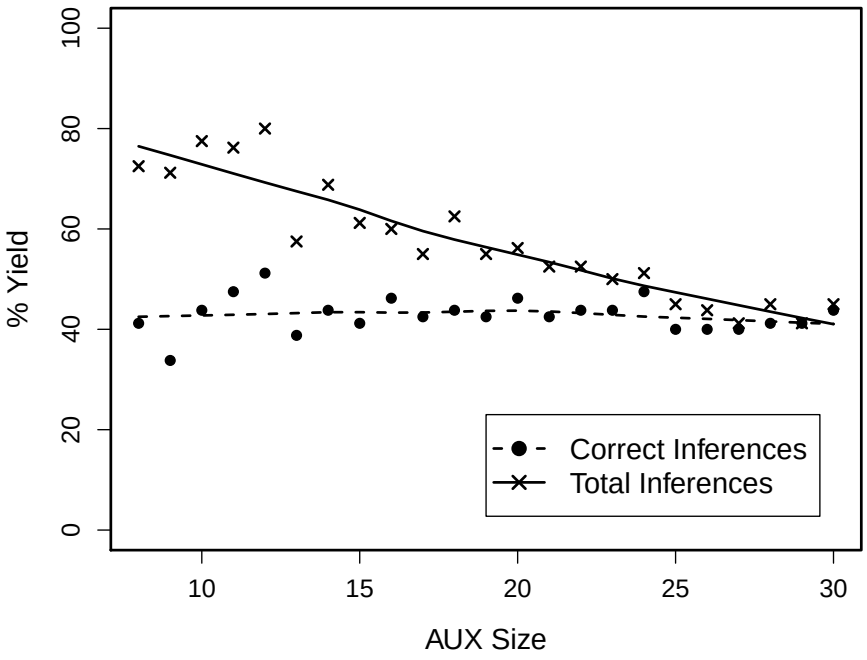


Figure 5.3: Hunch: Yield vs. size of AUX for simulated users. $threshold_{score}$ is 70%, $threshold_{support}$ is 51.25% of AUX size.

expected, the inference algorithm performs better on less active questions. Overall, our algorithm achieves 78% accuracy with 100% yield.

Figure 5.3 shows, for a particular setting of parameters, how yield and accuracy vary with the size of auxiliary information. As expected, larger AUX reduces the number of incorrect inferences, resulting in higher accuracy.

Inference vs. prediction. To illustrate the difference between inference and prediction, we compared the performance of our algorithm to a Bayesian predictor. Let X_i be a possible answer to a TARGET question and Y_j be the known response to the jth AUX question. The predictor selects X_i to maximize $P(X_i|Y_1)P(X_i|Y_2)\cdots P(X_i|Y_n)$, where $P(X_i|Y_j) = \frac{P(X_iY_j)}{P(Y_j)} = \frac{covar(X_i,Y_j)+P(X_i)P(Y_j)}{P(Y_j)}$. Individual probabilities and covariances are available from the Hunch API. As Figure 5.4 shows, our algorithm greatly

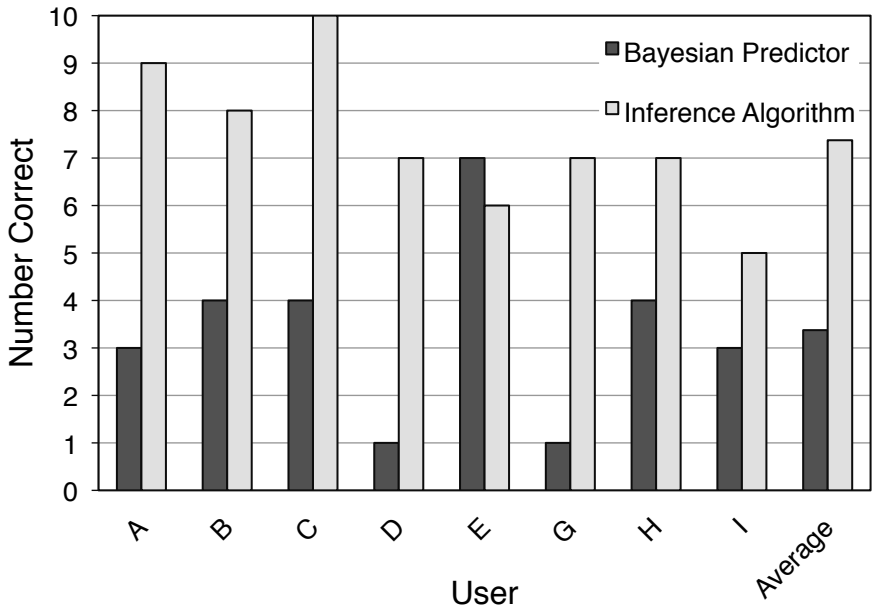


Figure 5.4: Inference vs. Bayesian prediction on simulated Hunch users. $threshold_{score}$ is 55%, $threshold_{support}$ is 40% of AUX size. We used low thresholds to ensure that our algorithm reached 100% yield.

outperforms the Bayesian predictor, averaging 73.75% accuracy vs. 33.75% accuracy for the predictor. This is not surprising, because our algorithm is not making educated guesses based on what similar users have done—it is observing the effects of actual transactions!

5.5.2 LibraryThing

LibraryThing is an online recommender system for books with over 1 million members and 55 million books in their online "libraries" [62]. For each book, LibraryThing provides a "People with this book also have... (more common)" list and a "People with this book also have... (more obscure)" list. These lists have ten to thirty items and are updated daily.

Algorithm 4 shows our inference procedure. Here delta matrices contain changes in "related books (common)" and "related books (obscure)" lists. Because these lists

Algorithm 4: LIBRARYTHINGINFERENCE

Input: set of "common" delta matrices $\mathcal{N}_C$, set of "obscure" delta matrices $\mathcal{N}_O$, scoring function: $\mathbb{R}^{|\mathcal{A}|} \to \mathcal{P}(\mathcal{A})$, set of AUX items $\mathcal{A}$, lookup table of popularities P, $threshold_{pop}$ for popularity of AUX books, $threshold_{score}$ for score, *window* time interval in which we expect changes to propagate

Output: Inferred books from the target user's library

```
inferences = {}
scoreSets = dict{}
foreach observation time τ do
    foreach delta matrix N_Δ in N_C, N_O within window of τ do
        foreach target item t in N_Δ do
            if t ∉ scoreSets then
                scoreSets[t] = {}
            scoreSets[t] = scoreSets[t] ∪ SCOREFUNCTION(N_Δ[t], A, P, threshold_pop)
foreach target item t in keys of scoreSets do
    if |scoreSets[t]| ≥ threshold_score and t ∉ A then
        inferences = inferences ∪ {t}
return inferences
```

Algorithm 5: LTSCOREFUNCTION

Input: Delta-matrix row $\mathbb{T}_t$ corresponding to book t, implemented as a dictionary keyed on AUX books and containing the relative change in list position, set of AUX books $\mathcal{A}$, lookup table of popularities P, *threshold* for popularity of AUX book

Output: Subset of AUX books with which t increased in correlation

```
scoreSet = {}
foreach a ∈ A do
    if popularity P[a] ≤ threshold then
        if T_t[a] > 0 then
            scoreSet = scoreSet ∪ {a}
return scoreSet
```

are updated frequently, the algorithm takes a *window* parameter. When a book is added to the user's library, we expect that, within *window*, it will rise in the lists associated with the "auxiliary" books. Algorithm 5 counts the number of such lists, provided that they are associated with less popular auxiliary books.

We used the site's public statistics to select three users who had consistently high activity, with thirty to forty books added to their collections per month. Two-hundred random books from each user's collection were selected as auxiliary information. The experiment ran for four months. Results are shown in Figure 5.5.

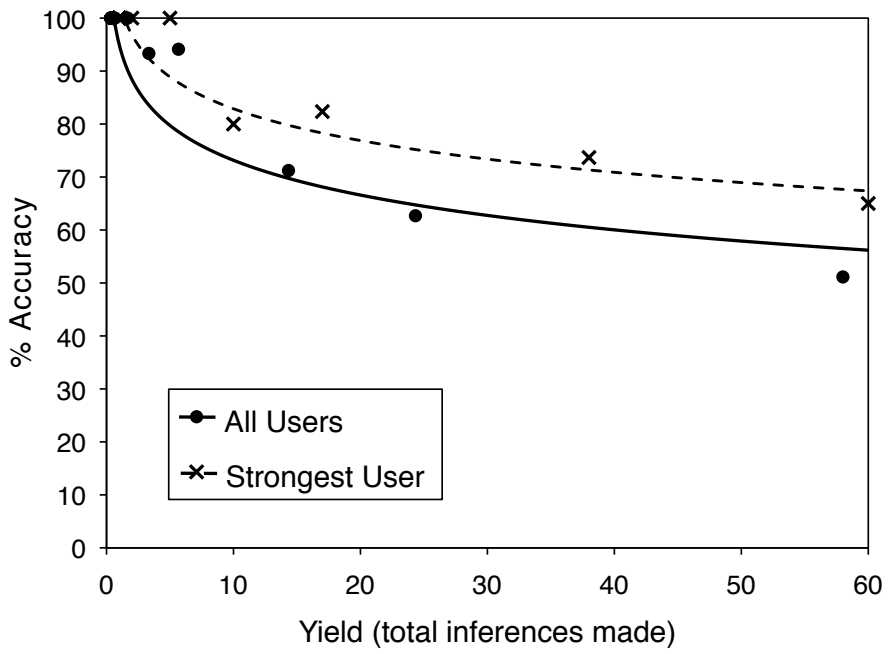


Figure 5.5: LibraryThing: Accuracy vs. yield: for all users (averaged) and for the strongest user.

When parameters are tuned for higher yield, related-items inference produces many false positives for LibraryThing. The reason is that many public libraries have accounts on LibraryThing with thousands of books. Even if some book rises in K lists related to the target user's library, K is rarely more than twenty, and twenty books may not be enough to identify a single user. False positives often mean that another LibraryThing user, whose library is a superset of the target user's library, has added the inferred book to their collection.

5.5.3 Last.fm

Last.fm is an online music service with over thirty-million monthly users, as of March 2009 [59]. Last.fm offers customized radio stations and also records every track that a user listens to via the site or a local media player with a Last.fm plugin. A user's listening history is used to recommend music and is public by default. Users may choose

not to reveal real-time listening data, but Last.fm automatically reveals the number of times they listened to individual artists and tracks. Public individual listening histories provide a "ground-truth oracle" for this experiment, but the implications extend to cases where individual records are private.

Aggregated data available through the Last.fm site and API include user-to-user recommendations ("neighbors") and item-to-item recommendations for both artists and tracks. We focus on track-to-track recommendations. Given a track, the API provides an ordered list of up to 250 most similar tracks along with "match scores," a measure of correlation that is normalized to yield a maximum of one in each list.

Our Last.fm experiment is similar to our LibraryThing experiment. Using changes in the similarity lists for a set of known auxiliary (AUX) tracks, we infer some of the TARGET tracks listened to by the user during a given period of time.

Unlike other tested systems, Last.fm updates all similarity lists simultaneously on a relatively long cycle (typically, every month). Unfortunately, our experiment coincided with changes to Last.fm's recommender system.[4] Thus during our six-month experiment, we observed a single update after approximately four months. Batched updates over long periods make inference more difficult, since changes caused by a single user are dominated by those made by millions of other users.

Several features of Last.fm further complicate adversarial inference. Last.fm tends not to publish similarity lists for unpopular tracks, which are exactly the lists on which a single user's behavior would have the most impact. The number of tracks by any given artist in a similarity list is limited to two. Finally, similarity scores are normalized, hampering the algorithm's ability to compare them directly.

Setup. We chose nine Last.fm users with long histories and public real-time listening information (to help us verify our inferences). As auxiliary information, we used the

[4]Private communication with Last.fm engineer, July 2010.

500 tracks that each user had listened to most often, since this or similar information is most likely to be available to an attacker.

Inferring tracks. After each update, we make inferences using the generic algorithm for related-items lists (Algorithm 1) with one minor change: we assume that each value in N_Δ also stores the initial position of the item prior to the change.

Our scoring function could simply count the number of AUX similarity lists in which a TARGET track has risen. To strengthen our inferences, we use the scoring function shown in Algorithm 6. It takes into account information summarized in Table 5.1 (these values are computed separately for the lists in which the TARGET rose and those in which it fell, for a total of sixteen values). To separate the consequences of individual actions from larger trends and spurious changes, we consider not only the number of similarity lists in which the TARGET rose and fell, but also the TARGET's average starting position and the average magnitude of the change. Because the expected impact of an individual user on an AUX item's similarity list depends on the popularity of the AUX item, we consider the average number of listeners and plays for AUX tracks as well.

We also consider information related to the clustering of AUX tracks. Users have unique listening patterns. The more unique the combination of AUX tracks supporting an inference, the smaller the number of users who could have caused the observed changes. We consider three properties of AUX supports to characterize their uniqueness. First, we look at whether supports tend to be by the same artist. Second, we examine whether supports tend to appear in each other's similarity lists, factoring in the match scores in those lists. Third, the API provides weighted tag vectors for tracks, so we evaluate the similarity of tag vectors between supports.

To fine-tune the parameters of our inference algorithm, we use machine learning. Its purpose here is to understand the Last.fm recommender system itself and is thus very different from collaborative filtering—see Section 5.4. For each target user, we

Algorithm 6: LastFmScoreFunction

Input: Delta-matrix row $\mathbb{T}_t$ corresponding to item t. Each vector entry contains *initPos* - the initial position of t in the list, and *change* - the number of places t has risen or fallen in the list. We also assume that each AUX item is associated with (*listeners*, *plays*, *simList*, *tags*) which are, respectively, the number of listeners, the number of plays, the updated similarity list with match scores, and the vector of weights for the item's tags.

Output: A *score* for the row

$score = 0$
$\mathcal{A}_{rise} = \{a \in \mathbb{T}_t \text{ such that } \mathbb{T}_t[a][change] > 0\}$
$\mathcal{A}_{fall} = \{a \in \mathbb{T}_t \text{ such that } \mathbb{T}_t[a][change] < 0\}$
$data_{rise} = data_{fall} = [0, 0, 0, 0, 0, 0, 0, 0]$
foreach $relevant \in \{rise, fall\}$ **do**
  $numSupports = |\mathcal{A}_{relevant}|$
  $avgInitPos = avg(\mathbb{T}_t[a][initPos] \text{ for } a \in \mathcal{A}_{relevant})$
  $avgChange = avg(\mathbb{T}_t[a][change] \text{ for } a \in \mathcal{A}_{relevant})$
  $avgListeners = avg(a[listeners] \text{ for } a \in \mathcal{A}_{relevant})$
  $avgPlays = avg(a[plays] \text{ for } a \in \mathcal{A}_{relevant})$
  $avgArtistScore = avgSimScore = avgTagScore = 0$
  foreach *aux item* $a \in \mathcal{A}_{relevant}$ **do**
    foreach *aux item* $a' \neq a$ *in* $\mathcal{A}_{relevant}$ **do**
      if $a[artist] == a'[artist]$ **then**
        $avgArtistScore += 1$
      if a' *in* $a[simList]$ **then**
        $simList = a[simList]$
        $avgSimScore += simList[a'][matchScore]$
      $tags = norm(a[tags])$
      $tags' = norm(a'[tags])$
      $avgTagScore += cosineSim(tags, tags')$
  if $numSupports > 0$ **then**
    $avgArtistScore = avgArtistScore / numSupports$
    $avgSimScore = avgSimScore / numSupports$
    $avgTagScore = avgTagScore / numSupports$
    $data_{relevant} = [numSupports, avgInitPos, avgChange, avgListeners, avgPlays, avgArtistScore, avgSimScore, avgTagScore]$
if $|\mathcal{A}_{rise}| > 0$ **then**
  $score = \text{MLProbCorrect}(data_{rise}, data_{fall})$
return *score*

train the PART learning algorithm [41] on all other target users, using the sixteen features and the known correct or incorrect status of the resulting inference. The ability to analyze data from multiple users is not necessary: a real attacker can train on auxiliary data for the target user over multiple update cycles or use his knowledge

Table 5.1: Attributes of candidate inferences. Computed separately for AUX items for which TARGET rose and those for which it fell.

Attribute	Definition
numSupports	Number of AUX similarity lists in which TARGET rose/fell
avgInitPos	Average initial position of TARGET item in supporting AUX item similarity list
avgChange	Average change of TARGET item in supporting AUX item similarity lists
avgListeners	Average number of listeners for AUX items supporting TARGET
avgPlays	Average number of plays for AUX items supporting TARGET
avgArtistScore	Average number of other AUX supports by same artist as any given AUX support
avgSimScore	Average sum of match scores for all other AUX supports appearing in any given AUX support's updated similarity list
avgTagScore	Average sum of cosine similarity between normalized tag weight vectors of any given AUX support and every other AUX support

of the recommender system to construct a model without training. The model obtained by training on other, possibly unrepresentative users may underestimate the power of an attacker.

After constructing the model, we feed the sixteen features of each TARGET into the PART algorithm using the WEKA machine-learning software [46]. Given the large number of incorrect inferences in the test data, WEKA conservatively classifies most attempted inferences as incorrect. WEKA also provides a probability that an inference is correct (represented by MLProbCorrect in Algorithm 6). We take these probabilities into account when scoring inferences.

We also account for the fact that similarity lists change in length and cannot contain more than two tracks by the same artist. For example, if a track was "crowded out" in a similarity list by two other tracks from the same artist, we know that its position was below the lowest position of these two tracks. Therefore, we judge the magnitude of any rise relative to this position rather than the bottom of the list. Similarly, if a list grows, we can confirm that a track has risen only if it rises above the bottom position in the previous list.

Results. The performance of our inference algorithm was negatively impacted by the fact that instead of a typical monthly update, it had to make inferences from a huge, four-month update associated with a change in the recommender algorithm.[5] Figures 5.6 and 5.7 show the results for two sample users; different points correspond to different settings of the threshold parameter of the algorithm. For the user in Figure 5.6, we make 557 correct inferences at 21.3% accuracy (out of 2,612 total), 27 correct inferences at 50.9% accuracy, and 7 correct inferences at 70% accuracy. For the user in Figure 5.7, we make 210 correct inferences at 20.5% accuracy and 31 correct inferences at 34.1% accuracy.

[5]Last.fm's changes to the underlying recommender algorithm during our experiment may also have produced spurious changes in similarity lists, which could have had an adverse impact.

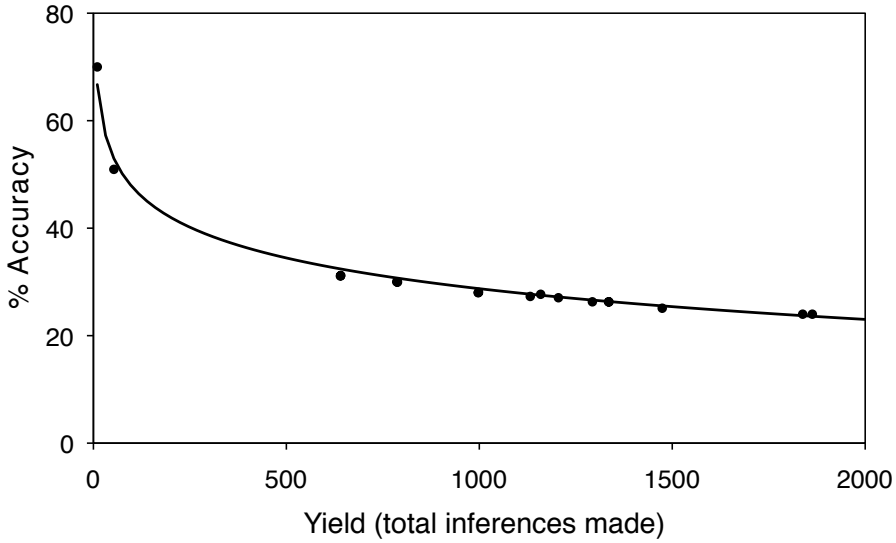


Figure 5.6: Accuracy vs. yield for an example Last.fm user.

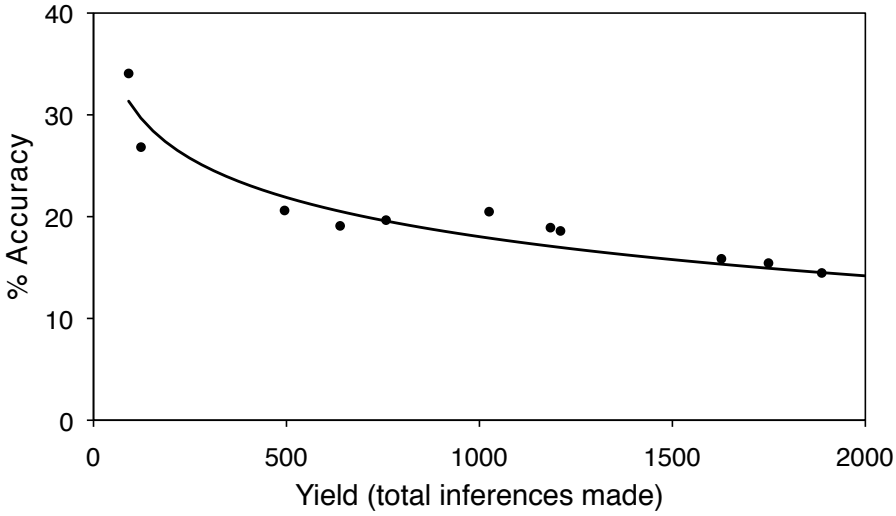


Figure 5.7: Accuracy vs. yield for another Last.fm user.

For a setting at which 5 users had a minimum of 100 correct inferences, accuracy was over 31% for 1 user, over 19% for 3 users, and over 9% for all 5 users. These results suggest that there exist classes of users for whom high-yield and moderate-accuracy inferences are simultaneously attainable.

5.5.4 Amazon

We conducted a limited experiment on Amazon's recommender system. Without access to users' records, we do not have a "ground-truth oracle" to verify inferences (except when users publicly review an inferred item, thus supporting the inference). Creating users with artificial purchase histories would have been expensive and difficult to replicate, and the user set would not have been representative of Amazon users.

The primary public output of Amazon's recommender system is "Customers who bought this item also bought ..." item similarity lists, typically displayed when a customer views an item. Amazon reveals each item's sales rank, which is a measure of the item's popularity within a given category.

Amazon customers may review items, and there is a public list of tens of thousands of "top reviewers," along with links to their reviews. Each reviewer has a unique reviewer identifier. Reviews include an item identifier, date, and customer opinions expressed in various forms. Customers are not required to review items that they purchase and may review items that they did not purchase from Amazon.

Setup. Amazon allows retrieval of its recommendations and sales-rank data via an API. The data available via the API are only a subset of that available via the website: only the 100 oldest reviews of each customer (vs. all on the website) and only the top 10 similar items (vs. 100 or more on the website).

We chose 999 customers who initially formed a contiguous block of top reviewers outside the top 1,000. We used the entire set of items previously reviewed by

each customer as auxiliary information. The average number of auxiliary items per customer varied between 120 and 126 during our experiment. Note that this auxiliary information is imperfect: it lacks items that the customer purchased without reviewing and may contain items the customer reviewed without purchasing.

Data collection ran for a month. We created a subset of our list containing active customers, defined as those who had written a public review within six months immediately prior to the start of our experiment (518 total). If a previously passive reviewer became active during the experiment, we added him to this subset, so the experiment ended with 539 active customers. For each auxiliary item of each active customer, we retrieved the top ten most-related items (the maximum permitted by the API)[6] daily. We also retrieved sales-rank data for all items on the related-item lists.[7]

Making inferences. Our algorithm infers that a customer has purchased some target item t during the observation period if t enters or rises in the related-items lists associated with at least K auxiliary items for the customer. We call the corresponding auxiliary items the supporting items for each inference. The algorithm made a total of 290,182 unique (user, item) inferences based on a month's worth of data; of these, 787 had at least five supporting items.

One interesting aspect of Amazon's massive catalog and customer base is that they make items' sales ranks useful for improving the accuracy of inferences. Suppose (case 1) that you had previously purchased item A, and today you purchased item B. This may have the same effect on their related-items lists as if (case 2) you had previously purchased B and today purchased A. Sales rank can help distinguish between these two cases as well as more complicated varieties. We expect the sales rank for most

[6]The set of inferences would be larger (and, likely, more accurate) for an attacker willing to scrape complete lists, with up to 100 items, from the site.

[7]Because any item can move into and off a related-items list, we could not monitor the sales ranks of all possible target items for the full month. Fortunately, related-items lists include sales ranks for all listed items.

items to stay fairly consistent from day to day given a large number of items and customers. Whichever item was purchased today, however, will likely see a slight boost in its sales rank relative to the other. The relative boost will be influenced by each item's popularity, *e.g.*, it may be more dramatic if one of the items is very rare.

Case studies. Amazon does not release individual purchase records, thus we have no means of verifying our inferences. The best we can do is see whether the customer reviewed the inferred item later (within 2 months after the end of our data collection). Unfortunately, this is insufficient to measure accuracy. Observing a public review gives us a high confidence that an inference is correct, but the lack of a review does not invalidate an inference. Furthermore, the most interesting cases from a privacy perspective are the purchases of items for which the customer would not post a public review.

Therefore, our evidence is limited to a small number of verifiable inferences. We present three sample cases. Names and some details have been changed or removed to protect the privacy of customers in question. To avoid confusion, the inferred item is labeled t in all cases, and the supporting auxiliary items are labeled a_1, a_2, and so on.

Mr. Smith is a regular reviewer who had written over 100 reviews by Day 1 of our experiment, many of them on gay-themed books and movies. Item t is a gay-themed movie. On Day 20, its sales rank was just under 50,000, but the ranking jumped to under 20,000 by Day 21. Mr. Smith's previous reviews included items a_1, a_2, a_3, a_4, and a_5. Item t was not in the similarity lists for any of them on Day 19 but had moved into the lists for all five by Day 20. Based on this information, our algorithm inferred that Mr. Smith had purchased item t. Within a month, Mr. Smith reviewed item t.

Ms. Brown is a regular reviewer who had commented on several R&B albums in the past. Item t is an older R&B album. On Day 1, its rank was over 70,000, but the

rank decreased to under 15,000 by Day 2. Ms. Brown had previously reviewed items a_1, a_2, and a_3, among others. Item t moved into item a_1 and item a_2's similarity lists on Day 2 and also rose higher in item a_3's list that same day. Based on this information, our algorithm inferred that Ms. Brown had purchased item t. Within two months, Ms. Brown reviewed item t.

Mr. Grant is a regular reviewer who appears to be interested in action and fantasy stories. Item t is a fairly recent fantasy-themed movie. On Day 18, its sales rank jumped from slightly under 35,000 to under 15,000. It experienced another minor jump the following day, indicating another potential purchase. Mr. Grant's previous reviews included items a_1, a_2, and a_3. On Day 19, item t rose in the similarity lists of a_1 and a_2 and entered a_3's list. None of the supporting items had sales rank changes that indicate purchases on that date. Based on this information, our algorithm inferred that Mr. Grant had bought item t. Within a month, Mr. Grant reviewed item t.

In all cases, the reviewers are clearly comfortable with public disclosure of their purchases since they ultimately reviewed the items. Nevertheless, our success in these cases suggests a realistic possibility that sensitive purchases can be inferred. While these examples include inferences supported by items in a similar genre, we have also observed cross-domain recommendations on Amazon, and much anecdotal evidence suggests that revealing cross-domain inferences are possible. For example, Fortnow points to a case in which an Amazon customer's past opera purchases resulted in a recommendation for a book of suggestive male photography [40].

5.6 Evaluation on a Simulated System

To test our techniques on a larger scale than is readily feasible with the real-world systems that we studied, we performed a simulation experiment. We used the Net-

flix Prize dataset [79], which consists of 100 million dated ratings of 17,770 movies from 460,000 users entered between 1999 and 2005. For simplicity, we ignored the actual ratings and only considered whether a user rated a movie or not, treating the transaction matrix as binary. We built a straightforward item-to-item recommender system in which item similarity scores are computed according to cosine similarity. This is a very popular method and was used in the original published description of Amazon's recommender system [64].

We restricted our simulation to a subset of 10,000 users who have collectively made around 2 million transactions.[8] Producing accurate inferences for the entire set of 460,000 users would have been difficult. Relative to the total number of users, the number of items in the Netflix Prize dataset is small: 17,770 movies vs. millions of items in (say) Amazon's catalog. At this scale, most pairs of items, weighted by popularity, have dozens of "simultaneous" ratings on any single day, making inference very challenging.

Methodology. We ran our inference algorithm on one month's worth of data, specifically July 2005. We assumed that each user makes a random 50% of their transactions (over their entire history) public and restricted our attention to users with at least 100 public transactions. There are around 1,570 such users, or 16%. These users together made around 1,510 transactions during the one-month period in question.

We assume that each item has a public similarity list of 50 items along with raw similarity scores that are updated daily. We also assume that the user's total number of transactions is always public. This allows us to restrict the attack to (customer, date) pairs in which the customer made five or fewer transactions. The reason for this is that some users' activity is spiky and they rate a hundred or more movies in a single day. This makes inference too easy because guessing popular movies at random would have a non-negligible probability of success. We focus on the hard case instead.

[8]Reported numbers are averages over 10 trials; in each trial, we restricted the dataset to a random sample of 10,000 users out of the 460,000.

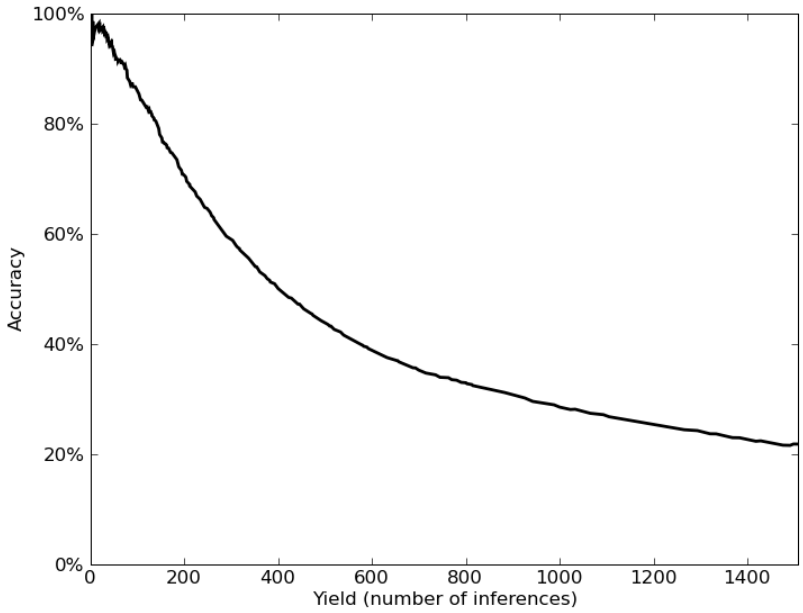


Figure 5.8: Inference against simulated recommender: yield vs. accuracy.

Results. The results of our experiment are summarized in Figure 5.8. The algorithm is configured to yield at most as many inferences as there are transactions: $1510/1570 \approx 0.96$ per vulnerable user for our one-month period. Of these, 22% are correct. We can trade off yield against accuracy by only outputting higher-scoring inferences. An accuracy of 80% is reached when yield is around 141, which translates to 7.5% of transactions correctly inferred. At an accuracy level of 90%, we can infer 4.5% of all transactions correctly. As mentioned earlier, these numbers are averages over ten trials.

Inference vs. prediction. Figure 5.9 shows that transactions involving obscure items are likely to be inferred in our simulated experiment. Figure 5.10 shows that our inference algorithm accurately infers even items that a predictor would rank poorly for a given user. For this graph, we used a predictor that maximizes the sum-of-cosines score, which is the natural choice given that we are using cosine similarity for the item-to-item recommendations. The fact that the median prediction rank of

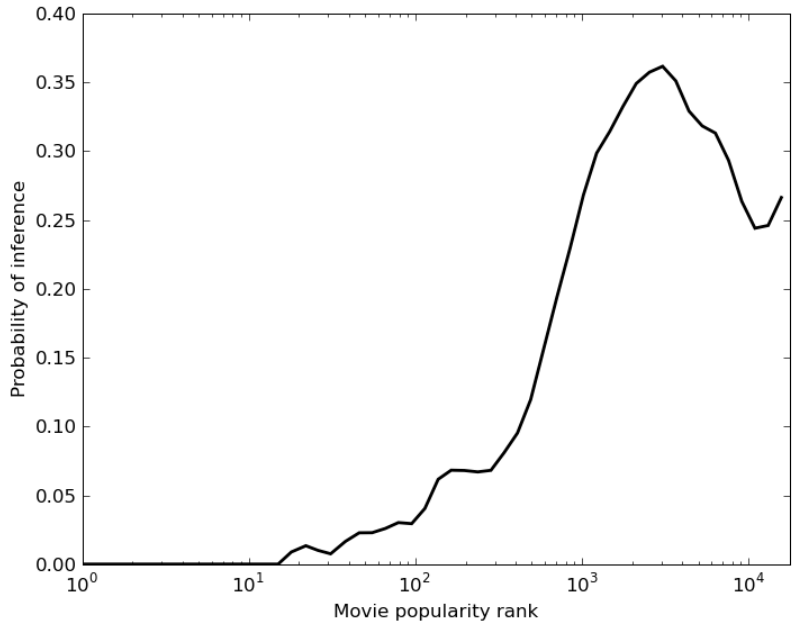


Figure 5.9: Likelihood of inference as a function of movie popularity.

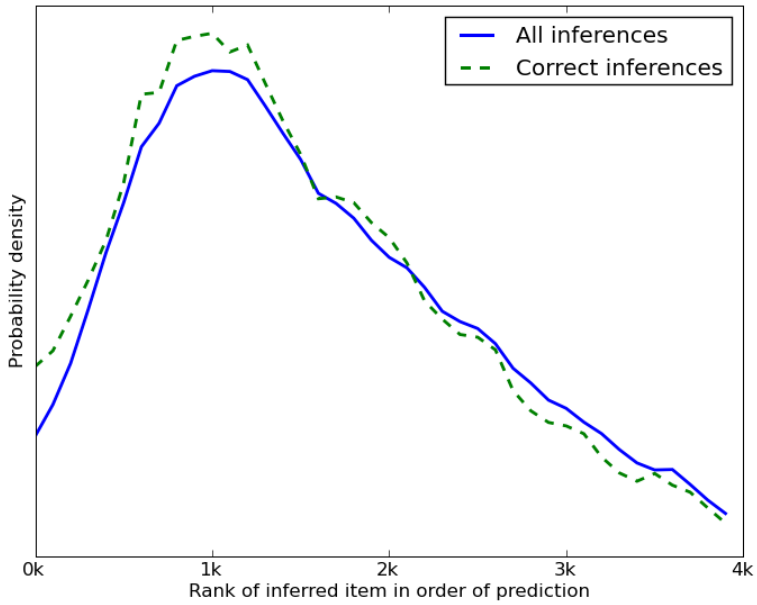


Figure 5.10: Simulated recommender: Inferences vs. predictions.

inferred items lies outside the top 1,000 means, intuitively, that we are not simply inferring the items that the user is likely to select anyway.

5.7 Mitigation

Differential privacy. Differential privacy is a set of algorithms and analytical techniques to quantify and reduce privacy risk to individual participants when answering statistical database queries [35]. Two threads of differential privacy research are potentially applicable to recommender systems. First, McSherry and Mironov showed how to construct differentially private covariance matrices, with some loss of accuracy for the collaborative filtering algorithms that use them [69]. While this is promising, they do not consider updates to the matrix, and it is not known how to build a dynamic system using this approach without destroying the privacy guarantee. The second line of research is on differential privacy under "continual observation," which directly addresses the needs of a dynamic system [37, 21]. These techniques break down, however, for systems with a large "surface area" such as similarity lists for every item. Designing a differentially private online recommender system remains an open problem.

Restrict information revealed by the system. Our inference algorithms exploit the fact that modern recommender systems make a large amount of information available for automated collection through their API. Restricting publicly available recommendation information may significantly complicate (but may not completely foil) the attack, while preserving legitimate functionality.

Limit the length of related-items lists. Amazon's "Customers who bought this item also bought ..." or Last.fm's "Similar music" lists can contain more than 100 items (although on Amazon, only the top 10 are available for automated collection via the API). Recommendations near the top of an item's related-items list have a strong

relationship with that item, which is unlikely to be impacted significantly by a single purchase. The ordering of recommendations near the bottom of the list is more volatile and reveals more information.

Factor item popularity into update frequency. Less popular items tend to be more identifying, so limiting the frequency of updates involving these items may decrease the efficacy of our inference attacks. For example, purchases of less popular items may be batched and reflected in the item-to-item matrix only several times a year. The same applies to their "sales rank," which for rare items can be sensitive to even a single purchase. Unfortunately, this mitigation technique may dull the impact of sudden, important shifts in the data, such as a surge in popularity of a previously obscure book.

Avoid cross-genre recommendations. In general, "straddlers," *i.e.*, customers with interests in multiple genres, tend to be at higher risk for privacy breaches. This risk could be mitigated by avoiding cross-genre recommendations except when items have strong relationships. This has the shortcoming of obstructing potentially surprising but useful recommendations.

Limit the speed and/or rate of data access. The large-scale, passive attacks described in this chapter require that the attacker extract a somewhat large quantity of data from the recommender system over a long period of time. Limiting the speed of access (by rate-limiting the API, using crawler-detection heuristics, *etc.*) may reduce the amount of available data and consequently the scale of the attack. Unfortunately, this may also prevent some legitimate uses of the data. Furthermore, these limits may not stop smaller, more focused attacks or a capable attacker (*e.g.*, one with access to a botnet).

It is hard to set a limit that would completely foil the attack. For example, Hunch's API limits each "developer key" to 5,000 queries per day (an attacker can

easily acquire multiple keys). Our experiments in Section 5.5.1 required between 500 and 2,500 queries per day per target user.

User opt-out. Many sites allow users to opt out of their recommender systems, but the opt-out is often incomplete. Amazon allows customers to request that certain purchases not be used for their own personalized recommendations. This option is primarily used to prevent gift purchases from influencing one's recommendations. For customers with privacy concerns, a more useful option would be to prevent a purchase from influencing the recommender system in any manner at all. If users habitually chose to opt out, however, recommendation quality could suffer significantly.

While each mitigation strategy has limitations, a careful combination of several techniques may provide substantial practical benefits with only modest drawbacks.

5.8 Related Work

Privacy and collaborative filtering. To our knowledge, this is the first project to show how to infer individual behavior from the public outputs of recommender systems. Previous work on privacy risks of recommenders focused on "straddlers" whose tastes span unrelated genres and assumed that the attacker is given the entire (supposedly anonymized) database of user transactions [87]. This model may be applicable in scenarios where collaborative filtering is outsourced, but is unrealistic for real-world recommender systems. Similarly, de-anonymization attacks require access to static datasets [42, 76].

Shilling attacks on collaborative filtering systems [72, 70] aim to influence the system by causing certain items to be recommended more often. We briefly mention an active attack on user-to-item collaborative filtering that is somewhat similar, but pursues a completely different goal.

Research on "social recommendations"—made solely based on a social graph—has shown that accurate recommendations necessarily leak information about the existence of edges between specific nodes in the graph [67]. This work differs from ours in that it (i) does not model user transactions, only edges in the social graph, (ii) does not consider temporal dynamics, and (iii) analyzes recommendations made to a user rather than public recommendations.

Previous work on protecting privacy in collaborative recommender systems aimed to hide individual user records from the system itself [19, 86, 92, 103]. These papers do not address the risk that individual actions can be inferred from temporal changes in the system's public recommendations and do not appear to provide much protection against this threat.

Privacy of aggregated data. Our attacks belong to a broad class of attacks that infer individual inputs from aggregate statistics. Disclosure of sensitive data from statistical summaries has long been studied in the context of census data [94]. Dinur and Nissim showed theoretically that an attacker who can query for arbitrary subsets of rows of a private database can learn the entire database even if noise has been added to aggregated answers [30]. Differential privacy was developed in part to provide a rigorous methodology for protecting privacy in statistical databases [33, 35]. Attacks on statistical databases often exploit the aggregates that happen to involve too few individuals. By contrast, we show that even with large aggregates, temporal changes can reveal underlying inputs.

Homer et al. showed that given a statistical summary of allele frequencies of a DNA pool—such as might be published in a genome-wide association study (GWAS)—it is possible to detect whether or not a target individual is represented in the pool, provided that the attacker has access to the individual's DNA [48]. The attack exploits the fact that DNA is very high-dimensional, thus the number of attributes is much greater than the number of records under consideration. Wang et al. strengthened

the attack of Homer et al. and also developed a second type of attack that uses a table of pairwise correlations between allele frequencies (also frequently published in GWA studies) to *disaggregate* the table into individual input sequences [98]. By contrast, the inference attacks described in this chapter are not based on disaggregation.

5.9 Conclusion

Recommender systems based on collaborative filtering have become an essential component of many websites. In this chapter, we showed that their public recommendations may leak information about the behavior of individual users to an attacker with limited auxiliary information. Auxiliary information is routinely revealed by users, but these public disclosures are under an individual's control: she decides which items to review or discuss with others. By contrast, item similarity lists and item-to-item covariances revealed by a recommender system are based on *all* transactions, including ones that users would not disclose voluntarily. Our algorithms leverage this to infer users' non-public transactions, posing a threat to privacy. We utilize aggregate statistics that contain no explicitly identifying information and are widely available from popular sites such as Hunch, Last.fm, LibraryThing, and Amazon. Our attacks are passive and can be staged by any user of the system. An active attacker can do even more.

We study larger, established sites as well as smaller and/or newer sites. Our results in the latter category are stronger, supporting the intuitive notion that customers of larger sites are generally safer from a privacy perspective and corroborating the findings in [69]. Smaller datasets increase the likelihood that individual transactions have a perceptible impact on the system's outputs.

Our work concretely demonstrates the risk posed by data aggregated from private records and undermines the widely accepted dichotomy between "personally identifi-

able" individual records and "safe," large-scale, aggregate statistics. Furthermore, it demonstrates that the dynamics of aggregate outputs constitute a new vector for privacy breaches. Dynamic behavior of high-dimensional aggregates like item similarity lists falls beyond the protections offered by any existing privacy technology, including differential privacy.

Modern systems have vast surfaces for attacks on privacy, making it difficult to protect fine-grained information about their users. Unintentional leaks of private information are akin to side-channel attacks: it is very hard to enumerate all aspects of the system's publicly observable behavior that may reveal information about individual users. Increasingly, websites learn from—and indirectly expose—aggregated user activity in order to improve user experience, provide recommendations, and support many other features. Our work demonstrates the inadequacy of current theory and practice in understanding the privacy implications of aggregated data.

Chapter 6

Conclusion

By its very nature, sensitive data is valuable. From our voting patterns to business trade secrets, we care about this data because its exposure could adversely impact anything from our financial interests to our reputation and personal safety. Nevertheless, we would forgo numerous benefits if we locked this data away and prevented any use beyond absolutely necessary ones. For example, an individual might not want to announce their interest in diabetes to the world, but that individual might wish to learn and help others learn which books on the topic are most popular and most highly rated. New ways of thinking and new technologies may make us more protective of this data or more willing to cede some control of it. Each case is ultimately a tradeoff, however, and individual preferences differ.

This work takes a step towards recognizing the unforeseen benefits and drawbacks posed by systems built on the use of sensitive data. A better understanding helps us in evaluating the tradeoffs in entrusting this data to others. We find that technological advances and novel insights may undermine even long-standing assumptions about the risks posed to individuals contributing sensitive records to a dataset (see Chapter 3 and Chapter 4). We also find that looking at privacy through the paradigm of anonymity alone may be insufficient. Even when a system outputs only complex

aggregate byproducts of sensitive data with no records to re-associate with individuals, we still may be able to infer sensitive details under real-world scenarios (see Chapter 5). The consequences of these results are sometimes undesirable but are not always so. For example, this work enables new election auditing approaches and novel methods for detecting cheating on standardized tests.

An important question is how we can more broadly detect and—in the detrimental cases—prevent unexpected exposure. In Chapter 2, we discuss a number of existing mitigation techniques, and we evaluate the efficacy of these techniques at multiple points throughout this thesis. A first step towards protecting sensitive data is understanding the desirable properties and known risks for systems using the data. A threat model can provide this for us.

Imagine the threat model for an anonymous survey conducted on bubble forms. One necessary property is protection of these forms against re-association with the corresponding individuals. Re-association could occur in a number of ways: invisible ink on the survey, paper fingerprinting, auxiliary data combined with the provided answers, bubble markings by the survey taker, etc. A threat model not only helps system developers uncover necessary properties and known risks but also provides a framework to prepare for the inevitable unexpected threats. The act of creating a threat model may even expose positive uses of data.

Given an understanding of the leakage risks posed by systems using sensitive data, a wide variety of mitigation techniques may help to reduce those risks. In some cases, procedural and other non-technical changes alone could achieve sufficient protection without undermining the quality of the system. In other cases, privacy-enhancing technologies of the type explored in Chapter 2 may be necessary to reduce the risks. Unfortunately, the scenarios in this thesis stretch beyond the limits of today's mitigation techniques.

Methods for scrubbing sensitive details from datasets or functions computed over them are relatively immature. We are only beginning to grasp what privacy means in a formal, mathematical sense and are still working to adapt these guarantees to new scenarios. The available techniques for achieving these guarantees tend to be fairly constrained primitives. Just as cryptography boasts a rich set of guarantees and techniques, however, data sanitization may eventually provide a bounty of definitions and building blocks for those looking to work responsibly with sensitive data.

General advancement in privacy-preserving data release methods should help protect against threats like those discussed in this dissertation. We briefly discuss two fruitful areas for future work: research on theoretically sound methods for the release of individual records and research on privacy-preserving collaborative filtering recommender systems.

Although many research uses of data involve computation of statistical queries over records in a dataset, release of scrubbed records can be more desirable than release of aggregate statistics alone. The former not only allows the party releasing data to perform less computational work but also may allow the party receiving the data to have greater flexibility in the analysis performed and tools used. Techniques such as k-anonymity [89] are built for this scenario but do not have the strong theoretical underpinnings of differential privacy [33, 35]. Differential privacy techniques can adapt to instances of this problem, as discussed in Chapter 2, but tend to be most useful and accurate for higher-level statistics. Additional research on techniques and limits in this area would be valuable. Ideally, techniques would handle internal data dependencies and provide utility guarantees for the dataset over certain popular classes of queries. We move must beyond the status quo in which released data is declared anonymous until someone proves otherwise and towards a situation in which we can make rigorous guarantees to the individuals put at risk by a data release.

Protecting sensitive transactions while enabling recommendations is also an interesting area for future work. Recommender systems are part of a broader class of systems that adapt to expressed user preferences. Search engines that adapt to user choices are a clear case of this, but sites that periodically revise their user experience based on logged data or products influenced by consumer surveys are also examples. In each case, someone is inferring what certain customers or end-users will prefer based on information about other individuals. As a result, advances in this area could be widely applicable.

Some research has been conducted on topics related to privacy-preserving recommendations—Chan et al. [21] were motivated in part by the work in Chapter 5—but we are still a long way from systems that handle the dynamics and high-dimensional data of collaborative filtering recommendations. We believe that additional research is justified. This work may even lead to changes in recommender systems themselves. For example, the issues associated with repeatedly computing functions over the same data raise the question of whether we could allow older data to go stale with little damage to utility and high payoffs for privacy. Ultimately, our goal is to maximize the benefits we derive from data by or about us without imposing the harms of leakage on the underlying individuals.

Bibliography

[1] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *TKDE*, 17(6), 2005.

[2] C. Aggarwal. On k-anonymity and the curse of dimensionality. In *31st International Conference on Very Large Data Bases*, 2005.

[3] R. Agrawal and R. Srikant. Privacy-preserving data mining. In *2000 ACM SIGMOD International Conference on Management of Data*, 2000.

[4] R. Agrawal, R. Srikant, and D. Thomas. Privacy preserving OLAP. In *24th ACM SIGMOD International Conference on Management of Data (SIGMOD 2005)*, 2005.

[5] Alameda County, California. Alameda County voting system demonstration. http://www.acgov/org/rov/votingsystemdemo.htm.

[6] A. W. Appel. Effective audit policy for voter-verified paper ballots in New Jersey, February 2007. http://www.cs.princeton.edu/~appel/papers/appel-nj-audits.pdf.

[7] J. A. Aslam, R. A. Popa, and R. L. Rivest. On auditing elections when precincts have different sizes. In *Proc. 2008 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '08)*.

[8] J. A. Aslam, R. A. Popa, and R. L. Rivest. On estimating the size and confidence of a statistical audit. In *Proc. 2007 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '07)*.

[9] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou R3579X? anonymized social networks, hidden patterns, and structural steganography. In *16th International World Wide Web Conference*, 2007.

[10] M. Barbaro and T. Zeller Jr. A face is exposed for AOL searcher no. 4417749. *New York Times*, August 9 2006.

[11] R. Bell, Y. Koren, and C. Volinsky. The BellKor solution to the Netflix Prize. http://www.netflixprize.com/assets/ProgressPrize2007_KorBell.pdf.

[12] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: The SuLQ framework. In *24th ACM SIGMOD-SIGACT-SICART Symposium on Principles of Database Systems (PODS 2005)*, 2005.

[13] A. Borges. Toward a new supermarket layout: From industrial categories to one stop shopping organization through a data mining approach. In *SMA Retail Symposium*, 2003.

[14] J. Brickell and V. Shmatikov. The cost of privacy: Destruction of data-mining utility in anonymized data publishing. In *Proc of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, August 2008.

[15] J. D. R. Buchanan, R. P. Cowburn, A.-V. Jausovec, D. Petit, P. Seem, G. Xiong, D. Atkinson, K. Fenton, D. A. Allwood, and M. T. Bryan. Forgery: 'fingerprinting' documents and packaging. *Nature*, 436:475, 2005.

[16] J. A. Calandrino, W. Clarkson, and E. W. Felten. Some consequences of paper fingerprinting for elections. In David Jefferson, Joseph Lorenzo Hall, and Tal Moran, editors, *Proceedings of EVT/WOTE 2009*. USENIX/ACCURATE/IAVoSS, August 2009.

[17] J. A. Calandrino, J. A. Halderman, and E. W. Felten. In defense of pseudorandom sample selection. In *Proc. 2008 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '08)*.

[18] J. A. Calandrino, J. A. Halderman, and E. W. Felten. Machine-assisted election auditing. In *Proc. 2007 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '07)*.

[19] J. Canny. Collaborative filtering with privacy. In *S & P*, 2002.

[20] R. Carback. How secret is your secret ballot? Part 1 of 3: Pattern voting. https://scantegrity.org/blog/2008/06/16/how-secret-is-your-secret-ballot-part-1-of-3-pattern-voting/, June 16 2008.

[21] H. Chan, E. Shi, and D. Song. Private and continual release of statistics. In *ICALP*, 2010.

[22] F. Chin. Security problems on inference control for sum, max, and min queries. In *Journal of the ACM*, 1986.

[23] W. Clarkson, T. Weyrich, A. Finkelstein, N. Heninger, J. A. Halderman, and E. W. Felten. Fingerprinting blank paper using commodity scanners. In *Proc. of IEEE Symposium on Security and Privacy*, May 2009.

[24] College Board. 2010 college-bound seniors results underscore importance of academic rigor. http://www.collegeboard.com/press/releases/213182.html.

[25] A. Cordero, D. Wagner, and D. Dill. The role of dice in election audits—extended abstract. In *IAVoSS Workshop on Trustworthy Elections 2006*.

[26] S. Coull, M. Collins, C. Wright, F. Monrose, and M. Reiter. On web browsing privacy in anonymized netflows. In *16th USENIX Security Symposium*, 2007.

[27] S. Coull, C. Wright, A. Keromytis, F. Monrose, and M. Reiter. Taming the devil: Techniques for evaluating anonymized network data. In *15th Annual Network and Distributed Systems Security Symposium*, 2008.

[28] S. Coull, C. Wright, F. Monrose, M. Collins, and M. Reiter. Playing devil's advocate: Inferring sensitive information from anonymized network traces. In *14th Annual Network and Distributed Systems Security Symposium*, 2007.

[29] M. Deshpande and G. Karypis. Item-based top-n recommendation algorithms. *TISSEC*, 22(1), 2004.

[30] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *PODS*, 2003.

[31] D. Dobkin, A. Jones, and R. Lipton. Secure databases: protection against user influence. In *ACM Transactions on Database Systems*, 1979.

[32] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–137, 2008.

[33] C. Dwork. Differential privacy. In *ICALP*, 2006.

[34] C. Dwork. Differential privacy. In *Proc of the 33rd International Colloquium on Automata, Language and Programming*, July 2006.

[35] C. Dwork. Differential privacy: A survey of results. In *TAMC*, 2008.

[36] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Third Theory of Cryptography Conference (TCC 2006)*, 2006.

[37] C. Dwork, M. Naor, T. Pitassi, and G. Rothblum. Differential privacy under continual observation. In *STOC*, 2010.

[38] A. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *22th ACM SIGMOD-SIGACT-SICART Symposium on Principles of Database Systems (PODS 2003)*, 2003.

[39] A. Feldman, J. A. Halderman, and E. W. Felten. Security analysis of the Diebold Accuvote-TS voting machine. In *Proc. 2007 USENIX/ACCURATE Electronic Voting Technology Workshop (EVT '07)*.

[40] L. Fortnow. Outed by Amazon. http://weblog.fortnow.com/2008/02/outed-by-amazon.html (Accessed Nov 17, 2010).

[41] E. Frank and I. H. Witten. Generating accurate rule sets without global optimization. In *ICML*, 1998.

[42] D. Frankowski, D. Cosley, S. Sen, L. Terveen, and J. Riedl. You are what you say: privacy risks of public mentions. In *SIGIR*, 2006.

[43] T. Gabriel. Cheaters find an adversary in technology. *New York Times*, December 27 2010.

[44] T. Gabriel. Under pressure, teachers tamper with tests. *New York Times*, June 10 2010.

[45] R. Garfinkel, R. Gopal, B. Pathak, R. Venkatesan, and F. Yin. Empirical analysis of the business value of recommender systems. http://ssrn.com/abstract=958770, 2006.

[46] S. R. Garner. WEKA: The Waikato environment for knowledge analysis. In *Proc. of the New Zealand Computer Science Research Students Conference*, pages 57–64, 1995.

[47] T. Greenson. Software glitch yields inaccurate election results. *Times-Standard*, December 5 2008.

[48] N. Homer, S. Szelinger, M. Redman, D. Duggan, W. Tembe, J. Muehling, J. Pearson, D. Stephan, S. Nelson, and D. Craig. Resolving individuals contributing trace amounts of DNA to highly complex mixtures using high-density SNP genotyping microarrays. *PLoS Genet*, 4, 2008.

[49] Humboldt County Election Transparency Project. http://www.humetp.org/.

[50] http://blog.hunch.com/?p=8264 (Accessed Nov 19, 2010).

[51] D. Irani, S. Webb, K. Li, and C. Pu. Large online social footprints–an emerging threat. In *CSE*, 2009.

[52] A. Jain, L. Hong, and S. Pankanti. Biometric identification. *Communications of the ACM*, 43(2):91–98, February 2000.

[53] K. C. Johnson. Election certification by statistical audit of voter-verified paper ballots, October 2004. http://papers.ssrn.com/sol3/papers.cfm?abstract_id=640943.

[54] I. T. Jolliffe. *Principal Component Analysis*. Springer, second edition, October 2002.

[55] A. M. Keller and D. Mertz. Privacy issues in an electronic voting machine. In *In Proceedings of the ACM Workshop on Privacy in the Electronic Society (WPES*, pages 33–34. ACM Press, 2004.

[56] K. Kenthapadi, N. Mishra, and K. Nissim. Simulatable auditing. In *Principles of Database Systems 2005*, 2005.

[57] T. Kohno, A. Stubblefield, A. Rubin, and D. Wallach. Analysis of an electronic voting system. In *Proc. 2004 IEEE Symposium on Security and Privacy*, pages 27–42.

[58] Tadayoshi Kohno, Andre Broido, , and K.C. Claffy. Remote physical device fingerprinting. In *IEEE Symposium on Security and Privacy*, 2005.

[59] http://blog.last.fm/2009/03/24/lastfm-radio-announcement (Accessed Nov 2, 2010).

[60] S. D. Levitt and S. J. Dubner. *Freakonomics: A Rogue Economist Explores the Hidden Side of Everything.* HarperCollins, 2006.

[61] N. Li, T. Li, and S. Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *23rd IEEE International Conference on Data Enginering*, 2007.

[62] http://www.librarything.com/press/ (Accessed Nov 10, 2010).

[63] G. Linden, J. Jacobi, and E. Benson. Collaborative recommendations using item-to-item similarity mappings. U.S. Patent 6266649. http://www.patentstorm.us/patents/6266649/fulltext.html, 2008.

[64] G. Linden, B. Smith, and J. York. Amazon.com recommendations: Item-to-item collaborative filtering. In *IEEE Internet Computing*, January-February 2003.

[65] Los Angeles County Registrar-Recorder / County Clerk. InkaVote Plus manual. http://www.lavote.net/voter/pollworker/PDFS/INKAVOTE_PLUS_HANDBOOK.pdf, 2011.

[66] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *ACM Transactions on Knowledge Discovery from Data*, 2007.

[67] A. Machanavajjhala, A. Korolova, and A. Sarma. Personalized social recommendations - accurate or private? Manuscript, 2010.

[68] C. McCutcheon. Absentee voting fosters trickery, trend's foes say. *Times Picayune*, October 2006.

[69] F. McSherry and I. Mironov. Differentially private recommender systems. In *KDD*, 2009.

[70] B. Mehta and W. Nejdl. Unsupervised strategies for shilling detection and robust collaborative filtering. *UMUAI*, 19(1–2), 2009.

[71] E. Metois, P. Yarin, N. Salzman, and J. R. Smith. FiberFingerprint identification. In *Proc. 3rd Workshop on Automatic Identification*, pages 147–154, 2002.

[72] B. Mobasher, R. Burke, R. Bhaumik, and C. Williams. Effective attack models for shilling item-based collaborative filtering systems. In *WebKDD*, 2005.

[73] M. Myagkov, P. C. Ordeshook, and D. Shakin. *The Forensics of Election Fraud: Russia and Ukraine*. Cambridge University Press, 2009.

[74] V. S. Nalwa. Automatic on-line signature verification. In *Proceedings of the Third Asian Conference on Computer Vision-Volume I - Volume I*, ACCV '98, pages 10–15, London, UK, 1997. Springer-Verlag.

[75] A. Narayanan and V. Shmatikov. How to break anonymity of the Netflix prize dataset. 2006.

[76] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *IEEE Symposium on Security and Privacy*, 2008.

[77] A. Narayanan and V. Shmatikov. De-anonymizing social networks. In *IEEE Symposium on Security and Privacy*, 2009.

[78] C. A. Neff. Election confidence: A comparison of methodologies and their relative effectiveness at achieving it, December 2003. http://www.votehere.net/papers/ElectionConfidence.pdf.

[79] http://www.netflixprize.com/rules (Accessed Nov 19, 2010).

[80] New Jersey Legislature. N.J.S.A. 19:4-13 (1976).

[81] E. Newton, L. Sweeney, and B. Malin. Preserving privacy by de-identifying facial images. *IEEE Transactions on Knowledge and Data Engineering*, 17:232–243, 2005.

[82] P. Ohm. Broken promises of privacy: Responding to the surprising failure of anonymization. *UCLA Law Review*, 57:1701–1777, 2010.

[83] Ruoming Pang, Mark Allman, Vern Paxson, and Jason Lee. The devil and packet trace anonymization. In *ACM SIGCOMM Computer Communication Review*, 2006.

[84] M. Peura and J. Iivarinen. Efficiency of simple shape descriptors. In *In Aspects of Visual Form*, pages 443–451. World Scientific, 1997.

[85] J. C. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines, 1998.

[86] H. Polat and W. Du. Privacy-preserving top-n recommendation on horizontally partitioned data. In *Web Intelligence*, 2005.

[87] N. Ramakrishnan, B. Keller, B. Mirza, A. Grama, and G. Karypis. Privacy risks in recommender systems. In *IEEE Internet Computing*, November-December 2001.

[88] R. G. Saltman. Effective use of computing technology in vote-tallying. Technical Report NBSIR 75-687, National Bureau of Standards, March 1975.

[89] P. Samarati and L. Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. In *Technical Report SRI-CSL-98-04, SRI Computer Science Laboratory*, 1998.

[90] T. S. Saponas, J. Lester, C. Hartung, S. Agarwal, and T. Kohno. Devices that tell on you: Privacy trends in consumer ubiquitous computing. In *16th USENIX Security Symposium*, 2007.

[91] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, 2001.

[92] R. Shokri, P. Pedarsani, G. Theodorakopoulous, and J-P. Hubaux. Preserving privacy in collaborative filtering through distributed aggregation of offline profiles. In *RecSys*, 2009.

[93] H. Stanislevic. Random auditing of e-voting systems: How much is enough?, August 2006. http://www.votetrustusa.org/pdfs/VTTF/EVEPAuditing.pdf.

[94] C. Sullivan. An overview of disclosure principles. U.S. Census Bureau Research Report, 1992.

[95] L. Sweeney. Simple demographics often identify people uniquely. *Carnegie Mellon University, Data Privacy Working Paper 3*, 2000.

[96] U.S. Department of Health & Human Services. IRB guidebook. http://www.hhs.gov/ohrp/irb/irb_guidebook.htm.

[97] Verified Voting Foundation. The verifier. http://www.verifiedvoting.org/verifier/.

[98] R. Wang, Y Li, X. Wang, H. Tang, and X. Zhou. Learning your identity and disease from research papers: Information leaks in genome wide association study. In *CCS*, 2009.

[99] Wisconsin Government Accountability Board. Spring 2011 election results. http://gab.wi.gov/elections-voting/results/2011/spring.

[100] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose, and G. M. Masson. Spot me if you can: Uncovering spoken phrases in encrypted VoIP conversations. In *IEEE Symposium on Security and Privacy*, 2008.

[101] C. V. Wright, L. Ballard, F. Monrose, and G. M. Masson. Language identification of encrypted VoIP traffic: Alejandra y Roberto or Alice and Bob? In *16th USENIX Security Symposium*, 2007.

[102] A. C.-C. Yao. Protocols for secure computations (extended abstract). In *FOCS 1982*, 1982.

[103] J. Zhan, C. Hsieh, I. Wang, T. Hsu, C. Liau, and D. Wang. Privacy-preserving collaborative recommender systems. In *SMC*, 2010.

[104] B. Zhu, J. Wu, and M. S. Kankanhalli. Print signatures for document authentication. In *Proc. 10th ACM Conference on Computer and Communications Security*, pages 145–154, 2003.