Privacy-preserving Messaging and Search: A Collaborative Approach

Giulia Fanti

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2015-230
http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-230.html

December 10, 2015

Privacy-preserving Messaging and Search: A Collaborative Approach

by

Giulia Cecilia Fanti

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering – Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Kannan Ramchandran, Chair
Professor Doug Tygar
Associate Professor Deirdre Mulligan

Fall 2015

Privacy-preserving Messaging and Search: A Collaborative Approach

Abstract

Privacy-preserving Messaging and Search: A Collaborative Approach

by

Giulia Cecilia Fanti

Doctor of Philosophy in Engineering – Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Kannan Ramchandran, Chair

In a free society, people have the right to consume and share public data without fear of retribution. However, today's technological landscape enables large-scale monitoring and censorship of networks by powerful entities (e.g., totalitarian governments); at worst, these entities may punish people for the information they consume or the opinions they espouse. This thesis considers two problems aimed at empowering people to freely share and consume public information: anonymous message spreading and privacy-preserving database search. In both areas, we present algorithmic innovations and analyze their correctness and efficiency. The key assumption underlying our work is that centralized architectures cannot reliably provide privacy-preserving services; not only are the incentive structures misaligned, but centralized infrastructures are often vulnerable to external breaches by hackers or government agencies, for example. We therefore restrict ourselves to distributed algorithms that rely on cooperation and resource-sharing between privacy-conscious individuals.

In the area of anonymous message spreading, we consider a user who wishes to spread a message to as many people as possible over an underlying connectivity network (e.g., a social network); this is the premise of Yik Yak, Whisper, and other popular anonymous messaging networks. Most existing social networks (anonymous or not) use a push-based mechanism to spread content to all of a user's neighbors on the contact network; if a neighbor approves the content by 'liking' it, this symmetric spreading propagates to the neighbor's neighbors, and so forth. Recent research suggests that under this spreading model, the true author of a message can be identified with non-negligible probability by a powerful global adversary. We propose an alternative, distributed spreading mechanism called *adaptive diffusion*, which breaks this symmetry. We show theoretically that adaptive diffusion gives optimal or asymptotically-optimal anonymity guarantees over certain classes of synthetic graphs for various adversarial models, while spreading nearly as fast as traditional symmetric mechanisms. On real-world graphs, we demonstrate empirically that adaptive diffusion gives significantly stronger anonymity properties than existing spreading mechanisms.

In the area of privacy-preserving search, we consider the foundations of a distributed, privacy-preserving search engine built over public data. Architecturally, we envision a peer-to-peer (P2P)

network in which each user stores a small piece of a public database; when a user wishes to search for something, she obtains it by requesting the information from her peer nodes, which execute a distributed search over the relevant data index. Critically, this operation should be privacy-preserving—that is, no peer node should learn anything about the contents of the user's query. Distributed search engines are not a new idea, but making such a service privacy-preserving and robust is algorithmically challenging.

One challenge is that if the database is changing over time and the network is not centrally controlled, distributed users may not have a unified view of the database. That is, some peers may be storing an *outdated* portion of the global database, causing existing private search and retrieval algorithms to fail. We introduce a distributed private retrieval algorithm that is robust to servers with similar, but not identical, views of the database, and show that it incurs asymptotically negligible overhead compared to traditional algorithms. Another challenge is that most search engine users submit queries with multiple keywords, and expect the result to contain all of the queried keywords; this is known as a *conjunctive* query. Existing distributed algorithms return results that contain *at least one* of the queried keywords. This approach can incur significant communication overhead, as well as computational overhead for the client, who must sort through the results. We propose a new privacy-preserving search algorithm that processes conjunctive queries while incurring a communication cost that scales linearly in the number of documents that contain *all* the queried keywords. Our private-search algorithms build on principles from distributed source coding, which permit us to reduce the communication cost by exploiting correlations between the data of distributed peer nodes.

To my family.

Contents

List of Figures

List of Tables

Acknowledgments

This dissertation would not have been possible without a number of people. First and foremost, I would like to thank my adviser, Professor Kannan Ramchandran. His support, guidance, and infinite patience have helped me grow so much over these last years, both professionally and personally. Kannan, thank you so much for this amazing opportunity to work with and learn from you; your intellectual curiosity, creativity, and ability to draw meaningful connections between topic areas never cease to amaze me.

I would like to sincerely thank my dissertation committee members, Professors Doug Tygar and Deirdre Mulligan. Doug, your candid advice and incisive questions helped me focus both the scope and content of this work. Deirdre, your insightful comments about the forces that motivate people—especially with regards to privacy—helped me to discard unrealistic mental models and think more clearly about user incentives.

I also want to extend heartfelt thanks to my mentors and collaborators at the University of Illinois Urbana-Champaign, Professors Pramod Viswanath and Sewoong Oh, who treated me as if I were one of their own students. Pramod, thank you for your unwavering support, for teaching me how to choose meaningful research problems, and for showing me when (and more importantly, how) to persevere on problems that seem impenetrable. Sewoong, thank you for your unflappable patience as I tried to keep up with your train of thought; learning from you is a true pleasure.

To Peter Kairouz (who contributed to a number of results in this thesis), Venkatesan Ekambaram, Babak Ayazifar, Matthieu Finiasz, Gerald Friedland, Barath Raghavan, Adam Lerner, Yahel Ben-David, Sebastian Benthall, Rachid El-Azouzi, Daniel Menasche, and the other amazing researchers I've had the pleasure of working with over the last five years, thank you for being fantastic collaborators; your talent, passion, and kindness are an inspiration. To my brilliant professors and teachers—both at Berkeley and beyond—thank you for sharing your knowledge and experiences. To my classmates and friends, thank you for your support and friendship. You made my time at Berkeley some of the best years of my life.

I gratefully acknowledge the NSF for funding this work with a Graduate Research Fellowship and grants CCF-0964018, CCF-30909, and CCF-30149. Thanks as well to Google for an internship with with my wonderful mentors, Vasyl Pihur, Úlfar Erlingsson, and Justin Ma.

Finally, I would like to thank my family, whose unconditional love and support made the lows bearable and the highs worthwhile. Mamma, Papà, Vic, Kevin—you mean the world to me, and I could never have done any of this without you.

Chapter 1

Introduction

In a free society, people have the right to consume and distribute information without being monitored or surveilled [39]. However, the typical modern Internet user browses the web, shops online, posts pictures, and sends messages while (often unknowingly) revealing intimate personal details to the government [72], external hackers [55], cellular and internet service providers [73], third parties buying access to user data, and of course service providers themselves (particularly search engines). This reality—which stems largely from technological constraints—has led to unprecedented levels of public monitoring and constitutes a major ongoing privacy violation. More problematically, when these privacy violations are orchestrated by powerful and corrupt entities (e.g., totalitarian governments), those entities may punish people for the information they consume or the opinions they share online [103].

Much thought has consequently gone into helping people utilize online services without sacrificing control of their personal information; proposed solutions often harness combinations of legal, economic, and technological tools. Unfortunately, regardless of approach, the notion of protecting user privacy is misaligned with the dominant economic and usability incentives of today's online economy: users want free, efficient services almost as much as service providers want users' data. Since people are ultimately unlikely to boycott services over privacy concerns [9], policy solutions that require service providers to relinquish data rights are unlikely to be sustainable. Similarly, services that require users to pay for privacy-preserving experiences are at an economic disadvantage. Nonetheless, a significant class of people—including political dissidents, whistleblowers, and other privacy-minded individuals—may be willing to tolerate higher costs (e.g., money, reduced efficiency) in order to privately access and participate in networks. The primary constraint is that no user should have to trust any other party with their data. Thus emerges a collaborative model of privacy, in which network users help one another achieve privacy, rather than relying on a benevolent or legally-bound central entity. This thesis works within such a framework of peer-faciliatated privacy—a concept that has been popular for decades in the privacy-preserving algorithms literature [22, 21, 92, 48, 35].

Information access and dissemination are fundamental operations in many networks (e.g., social networks, the Internet). Users cannot freely participate in such networks without enjoying privacy in both regards—sharing and consuming information. Building upon this collaborative

model of privacy, we ask two essential questions:

1. *Anonymous message spreading (dissemination):* How can we empower users to broadcast information or opinions without being personally linkable to that content, even by a powerful global adversary?

2. *Private search (access):* How can we empower users to search and retrieve content from a distributed data collection without revealing the content of that information to the network operator or external adversaries?

The answers to these questions could be integrated into a cohesive content-management system that provides anonymity to content-generators and privacy to users who access that content. However, we maintain that both problems are interesting of their own right, and we treat them as discrete problems in this thesis. For each problem, we propose novel algorithms and give provable guarantees on privacy, efficiency, and utility. In the remainder of this section, we provide some context for each problem of interest. We then give an outline of the thesis and list the main results presented in this work.

Anonymous message spreading

Microblogging platforms like Facebook and Twitter have become important Internet services. However, people who post controversial content on such platforms may face social, political, or economic repercussions, ranging from cyberbullying to physical punishment. In fact, there are documented cases of people being fired from their jobs or incarcerated for posting objectionable content online [98, 103]; this is especially true in countries under authoritarian rule. These realities reveal a legitimate need for microblogging platforms that protect author anonymity: this is the focus of the first portion of this thesis.

Informally, we treat a microblogging platform as a collection of users and an underlying connectivity network that constrains communication between users. This network could represent social connections, professional affiliations, or physical proximity; it is superimposed on an underlying communication network, e.g., the Internet. For instance, on Facebook, the connectivity network is the Facebook social graph. The user's goal on such a platform is to spread a message to as many other users as possible, while respecting the communication constraints of the underlying network (i.e., you can only transmit messages to your neighbors, and there is delay associated with each transmission). The adversary's goal is to learn who authored a given message, given the underlying contact network and incomplete metadata about the spread of the message.

A naive way to implement anonymous microblogging is to strip authorship metadata from every message. This approach is used in a recent crop of anonymous messaging social networks, like Whisper [3], Yik Yak [4], and the now-defunct Secret [2]. These applications allow a user to share messages with her peers without revealing the author's identity (Figure 1.1). However, these services are all *centralized*, so authorship information for every message is stored on central-

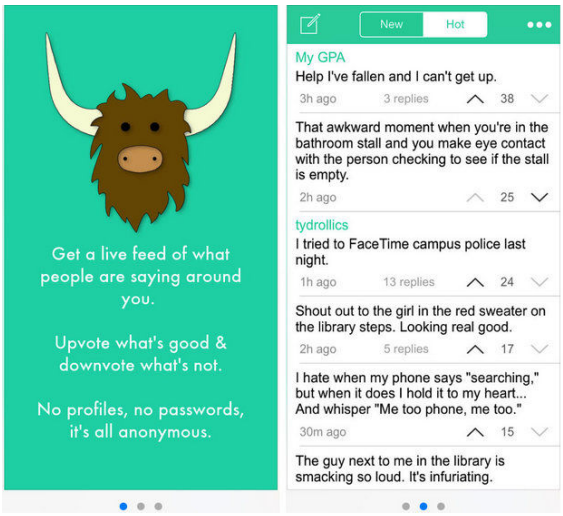


Figure 1.1: Popular anonymous messaging applications like Yik Yak [4] allow users to post content without any authorship information attached.

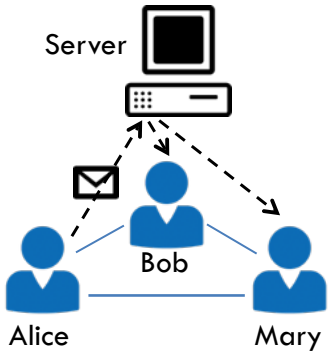


Figure 1.2: Existing anonymous messaging services are centralized, meaning that if Alice wants to broadcast an anonymous message to Bob and Mary, the message passes through a centralized server, which observes all authorship information.

ized servers that may be accessible to government agencies or hackers [67]. We are interested in architectures and algorithms that are truly anonymous—even to the service provider itself.

Even if anonymous messaging services like Secret or Yik Yak were not centralized, stripping authorship metadata would not protect the author against adversaries that can observe traffic patterns and participate in network activity in order to deanonymize users. In fact, recent work on rumor source identification [94, 93, 89, 46, 118, 36, 109] suggests that even when the adversary has a limited capability to observe message metadata in such a network, it can identify the true source with high accuracy. This weakness stems from the fact that existing microblogging applications use a symmetric, push-based content-spreading mechanism. That is, if a user composes or approves a message (e.g., by 'liking' it), the message gets propagated to all of the user's neighbors. This spreading model is often modeled mathematically by a *diffusion process*; the global symmetry of diffusion can be exploited (along with knowledge of the underlying connectivity network) to infer the true source of a message [94].

Given these weaknesses, our goal is to design *distributed* algorithms for disseminating messages with *provable* guarantees of message author anonymity, even against powerful global adversaries.

Private search

Another important class of privacy leaks results when users retrieve information from online repositories. When people query a search engine or a database, the contents of those queries may be sensitive [7]; if so, users may be targeted for extra monitoring or even censorship. For instance,

a user who types 'how to build a bomb' into a search engine could trigger additional monitoring, either by the search engine itself, or by an external law enforcement agency with access to search engine records. Offhand, this might seem like a good thing. However, it enables large-scale frameworks for surveillance and even censorship; when this monitoring surpasses legally-established limits (or limits set by the international community), the results can be damaging to society [5]. Even if users' queries are not particularly sensitive, when aggregated over time, they can enable an observer to build detailed, intimate profiles of people. The AOL search history data release is a prominent example of this [7].

Intuitively, this problem might seem fundamental: a user must reveal her query to a server in order to search a database stored on that server. Perhaps surprisingly, this is not the case; a recent body of research has investigated algorithms for searching and accessing a database without revealing the nature of users' queries to the server storing the database [23, 83]. Although this area has been studied for a long time, existing approaches are overwhelmingly inefficient, to the point that they have not found their way into usable products. In this portion of the thesis, the goal is to develop methods for searching and retrieving information from a database without revealing clients' queries to the server, while meeting strict efficiency constraints.

Outline and contributions

The problems studied in this thesis are not new. However, our approach differs dramatically from prior art, due to a combination of architectural decisions (i.e., a distributed architecture) and the algorithmic tools we use (i.e., adaptive, randomized algorithms and distributed source coding). Each of these seemingly small changes require a first-principles redesign of existing algorithms, and can have significant effects on the efficiency and functionality of existing algorithms. Each chapter therefore introduces a novel algorithmic solution to a distinct problem of interest. Because these chapters address somewhat different problems, the related work for this thesis is discussed on a per-chapter basis. An outline of the thesis is as follows:

Chapter 2: *(Anonymous Message Spreading)* This chapter considers the problem of anonymous broadcast messaging. In particular, we consider messaging applications built atop a social network, which constrains communication between users. We present *adaptive diffusion*: an algorithm that spreads content over a social network in such a way that even a powerful adversary with access to a message's spreading pattern cannot infer the message author. We consider two main adversarial models: the first adversary learns which users received a given message at a single, fixed point in time; we call this the *snapshot* adversary. For example, a snapshot adversary might represent a government agency that observes which users are present at a protest that was advertised on the social network. The protest attendees provide the adversary with a (noisy) snapshot of the network users that received the message. In the second adversarial model, the adversary takes the form of multiple, colluding "spy" nodes, which observe time-of-arrival metadata for each message; we call this the *spy-based* adversary. An example of a spy-based adversary would be a law-enforcement agency that creates fake accounts within a social network in order to monitor what traffic gets passed around at what time. Both of these adversaries use their observed information to infer the

true author of a given message. We describe these adversaries more precisely in Chapter 2.

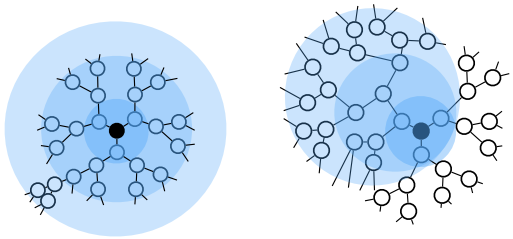


Figure 1.3: Spread of message under standard diffusion (left) and under adaptive diffusion (right).

Contributions:

- We describe the *adaptive diffusion* spreading protocol in detail, and prove that it exhibits optimal anonymity properties against a snapshot adversary when the underlying contact network is a regular tree. The key intuition behind adaptive diffusion is that it does not spread content symmetrically. Figure 1.3 illustrates the approach taken by most existing services, often modeled mathematically by diffusion (left panel). Adaptive diffusion (right panel) breaks the symmetry of diffusion by disseminating content quickly in some directions and slowly in others.
- We show that even under the stronger spy-based adversarial model, adaptive diffusion has asymptotically optimal hiding properties, in the degree of the underlying, regular tree.
- We demonstrate through simulation that when the underlying network is a real social graph (e.g., a subset of the Facebook social graph), adaptive diffusion spreads the message quickly and achieves nearly optimal hiding under both adversarial models.

Chapter 3: *(Private Information Retrieval in Unsynchronized Environments)* Privately *searching* data is closely related to the easier problem of privately *retrieving* data—a problem in which the address of the desired content is already known to the client (Figure 1.4). Understanding how to retrieve data privately is a critical building block in designing private search algorithms. Private information retrieval (PIR) algorithms enable a user to retrieve the wth record of a database without revealing w to the server. However, even this easier problem is not solved in a practical sense, especially in the critical use-case of distributed architectures.

The problems with distributed PIR arise in a subclass of schemes called multi-server PIR, which make use of multiple, non-colluding servers. Multi-server PIR algorithms are significant for two main reasons: *(a)* they are orders of magnitude more efficient than the corresponding

Figure 1.4: Our goal is to enable efficient, privacy-preserving search, in which the client's query remains unknown to the server. We tackle this by first considering the simpler problem of retrieval.

single-server PIR algorithms [81], and *(b)* they naturally fit in with our collaborative, distributed model of privacy. However, existing multi-server PIR schemes assume that each server possesses an identical copy of the database. This condition is likely to fail in distributed systems like peer-to-peer networks, where issues like stale content are commonplace due to mis-synchronization between nodes.

Thus, an important step in developing scalable private search algorithms is to first develop PIR algorithms that can handle stale data on distributed nodes. The literature does not provide any private retrieval solutions when servers do not store perfectly-identical, or *synchronized* databases.

Contributions:

- We propose the first multi-server PIR scheme to return the desired record even when servers' databases are not perfectly synchronized.
- We show both theoretically and through simulation that our scheme asymptotically has the same computational and communication complexity as state-of-the-art PIR schemes for synchronized databases; this comes at the expense of probabilistic success and two rounds of communication (most existing schemes require only one). Additionally, this approach efficiently processes multiple concurrent PIR queries.

Chapter 4: *(Efficient Private Search with Conjunctive Queries)* Armed with the robust private retrieval algorithm from Chapter 3, we revisit our overarching goal of private search. Private streaming search (PSS) algorithms allow users to conduct keyword queries on a collection of documents, without revealing those keywords to the server [83]. Although we are not explicitly interested in streaming applications, PSS algorithms are the main existing approach to privacy-preserving keyword searches over public data. A PSS algorithm returns a list of all documents in a collection containing the queried keyword(s). However, most existing PSS schemes are very inefficient due to reliance on computationally-heavy cryptographic tools, like homomorphic encryption [100, 81].

Although there has been some research on expanding PSS to multi-server architectures [24, 80], most work in this space has focused on single-server architectures [10, 49, 113, 82]. Meanwhile, a primary take-away message of the PIR literature is that multi-server architectures enable significant efficiency gains compared to single-server architectures. Similar gains in the PSS domain could

bring privacy-preserving keyword searches closer to reality. However, to harness these gains, there are practical, algorithmic question specific to keyword queries that have not yet been answered. For example, how does one process conjunctive keyword queries in a distributed architecture? In existing schemes, if a client enters more than one keyword (e.g., "Joseph Smith"), PSS algorithms return all documents that contain *at least one* of the keywords. The resulting client-side expense of identifying documents that contain all the desired keywords can be prohibitive. Given that most search engine queries are longer than one word [66], this is a significant barrier to the adoption of private search algorithms.

In this chapter, we lower these practical barriers by proposing distributed private search algorithms that are: *(a)* computationally more efficient than single-server cryptographic alternatives, and *(b)* able to handle practical, conjunctive queries. The algorithms in this chapter use PIR as a building block, but more fundamentally, they rely on the algorithmic application of distributed source coding introduced in Chapter 3.

Contributions:

- We present a private search algorithm that is designed to efficiently process multi-keyword, or *conjunctive*, queries. Notably, this method has a communication cost that scales with the number of documents with *all* the desired keywords; this has not been possible in prior work.

Chapter 5: *(Future Work and Conclusions)* We discuss some important directions for future research. We reiterate the contributions of this work and explain how they fit into the larger picture of enabling practical, privacy-preserving messaging and search.

Chapter 2

Anonymous Message Spreading

In this chapter, we study a basic building block of the messaging protocol that would underpin truly anonymous microblogging platform – *how to anonymously broadcast a single message on a contact network*, even in the face of a strong deanonymizing adversary with access to metadata. Specifically, we focus on anonymous microblogging built atop an underlying social network, such as a network of phone contacts or Facebook friends.

Adversaries

We consider three adversarial models, which capture different approaches to collecting metadata. In each case, the underlying contact network is modeled as a graph that is known to the adversary. Beyond knowing the underlying graph, the adversary could proceed in a few different ways:

The adversary might use side channels to infer whether a node is infected, i.e., whether it received the message. If an adversary collects only infection metadata for all network users, we call it a *snapshot* adversary. This could represent a state-level adversary that attends a Twitter-organized protest; it implicitly learns who received the protest advertisement, but not the associated metadata. The snapshot adversary is well-studied in the literature, primarily in the related problem of source identification [94, 109, 89, 46, 71].

Alternatively, the adversary might explicitly corrupt some fraction of nodes by bribery or coercion; these corrupted *spy nodes* could pass along metadata like message timestamps and relay IDs. If an adversary only collects information from spies, we call it a *spy-based* adversary. A spy-based adversary could represent a government agency participating in social media to study users, for instance. The adversary's reach may be limited by factors like account creation, contact network structure [31], or the cost of corrupting participants.

Finally, an adversary might employ a hybrid of these approaches. If an adversary uses spies and a snapshot, we call it a *spy+snapshot* adversary. This is the strongest adversarial model we consider.

This chapter is based on joint work with Peter Kairouz, Sewoong Oh, Kannan Ramchandran, and Pramod Viswanath [41, 40].

Spreading models

In social networks, messages are typically propagated based on users' approval, which is expressed via liking, sharing or retweeting. This mechanism, which enables social filtering and reduces spam, has inherent random delays associated with each user's time of impression and decision to "like" the message (or not). Standard models of rumor spreading in networks explicitly model such random delays via a *diffusion* process: messages are spread independently over different edges with a fixed probability of spreading (discrete time model) or an exponential spreading time (continuous time model). The designer can partially control the spreading rate by introducing artificial delays on top of the usual random delays due to users' approval of the messages.

We model this physical setup as a discrete-time system. At time $t = 0$, a single user $v^* \in V$ starts to spread a message over the contact network $G = (V, E)$ where users and contacts are represented by nodes and edges, respectively. Upon receiving the message, nodes approve it immediately. The assumption that all nodes are willing to approve and pass the message is common in rumor spreading analysis [94, 93, 118]. However, by assuming message approval is immediate, we abstract away the natural random delays typically modeled by diffusion. At the following timestep, the protocol decides which neighbors will receive the message, and how much propagation delay to introduce. Given this control, the system designer wishes to design a spreading protocol that makes message source inference difficult.

Specifically, after T timesteps, let $V_T \subseteq V$, G_T, and $N_T \triangleq |V_T|$ denote the set of infected nodes, the subgraph of G containing only V_T, and the number of infected nodes, respectively. at a given time T, the adversary uses all available metadata to estimate the source. We assume no prior knowledge of the source, so the adversary computes a maximum-likelihood (ML) estimate of the source $\hat{v}_{ML}$. We desire a spreading protocol that minimizes the probability of detection $P_D = \mathbb{P}(\hat{v}_{ML} = v^*)$.

Current state-of-the-art: Diffusion is commonly used to model epidemic propagation over contact networks. While simplistic (it ignores factors like individual user preferences), diffusion is a commonly-studied and useful model due to its simplicity and first-order approximation of actual propagation dynamics. Critically, it captures the symmetric spreading used by most social media platforms.

However, diffusion has been shown to exhibit poor anonymity properties; under the adversarial models we consider, the source can be identified reliably [94, 86]. We therefore seek a different spreading model with strong anonymity guarantees. We wish to achieve the following performance metrics:

(a) We say a protocol has an *order-optimal rate of spread* if the expected time for the message to reach n nodes scales linearly compared to the time required by the fastest spreading protocol.

(b) We say a protocol achieves a *perfect obfuscation* if the probability of source detection for the maximum likelihood estimator is order-optimal. The definition of optimality differs for different adversarial models, so we define this metric separately for each adversarial model.

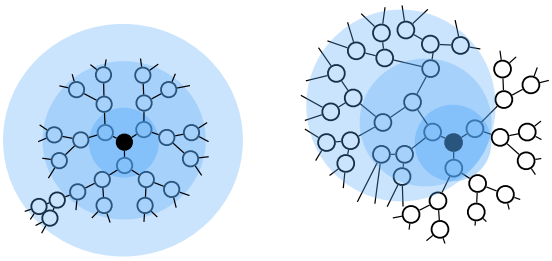


Figure 2.1: Illustration of a spread of infection when spreading immediately (left) and under adaptive diffusion (right).

Contributions

We introduce *adaptive diffusion*, a novel messaging protocol with provable author anonymity guarantees against all of the discussed adversarial models. Whereas diffusion spreads the message symmetircally in all directions, adaptive diffusion breaks that symmetry (Figure 2.1). This has different implications for different adversarial models, but it consistently yields stronger anonymity guarantees than diffusion. Adaptive diffusion is also inherently distributed and spreads messages fast, i.e., the time it takes adaptive diffusion to reach n users is at most twice the time it takes the fastest spreading scheme which immediately passes the message to all its neighbors.

We prove that over d-regular trees, adaptive diffusion provides perfect obfuscation of the source under the snapshot adversarial model. That is, the likelihood of an infected user being the source of the infection is equal among all infected users. Under the spy-based adversarial model, we prove that adaptive diffusion achieves optimal obfuscation, asymptotically in the degree of the underlying tree. For both adversarial models, we derive exact expressions for the probability of detection, and show they are optimal by providing a matching fundamental lower bound.

In practice, the contact networks are not regular infinite trees. For a general class of graphs which can be finite, irregular and have cycles, we provide results of numerical experiments on real-world social networks and synthetic networks showing that the protocol hides the source at nearly the best possible level of obfuscation under both adversarial models. Further, for a specific family of random *irregular* infinite trees, known as Galton-Watson trees, we characterize the probability of detection under adaptive diffusion. In the process, we prove a strong concentration for the extreme paths in the Galton-Watson tree that consists of nodes with large degrees, which might be of independent interest.

Finally, we show that even if the adversary has both types of metadata, as in the spy+snapshot adversary, the detection probability is close to that of weaker adversaries, which observe only one type of metadata.

Related Work

Anonymous broadcast messaging has been an active research area since the 1980s. The main thread of work in this area stems from the seminal dining cryptographers problem, and is solved by so-called DC-nets. We start by describing DC-nets, as well as some alternative approaches to anonymous broadcast messaging; we then explain why none of these approaches fully addresses our question of interest.

Dining cryptographer networks. Anonymous broadcast messaging was first studied by Chaum in his seminal work on the dining cryptographers problem [22]. The dining cryptographers' problem considers a scenario in which a group of cryptographers goes out to dinner, all seated around a table. After the meal, they wish to determine if one of them paid the bill, but for the sake of propriety, nobody should learn *who* paid the bill. Note that this problem setup is equivalent to that of a single user anonymously broadcasting a single bit, while the rest of the network participants stay silent.

Chaum's solution, commonly referred to as a dining cryptographer network, or a DC-net, uses pairwise shared secrets between users and the broadcast channel to compute an aggregate bit, which is 1 if any of the cryptographers paid the bill, and 0 otherwise [22]. This enables a single individual to broadcast a bit in such a way that the true author of the bit remains anonymous with information-theoretic guarantees.

The dining cryptographers' problem implicitly assumes the use of a complete connectivity graph, since each cryptographer shares a one-bit function output with every other participant. Notably, in order for one participant to broadcast a single bit anonymously, each non-communicating participant must also broadcast one bit to the entire network. In addition, each user must have shared, secret randomness with other users in the network. This introduces scalability challenges, which have impeded the development of large-scale systems for DC-net-based anonymous broadcasting. The dining cryptographers' adversarial model and problem setup were subsequently adopted in a number of works, with the primary goal of making DC-nets more scalable and robust to stronger adversaries. For example, the 'dining cryptographers in the disco' problem proposes schemes that have stronger guarantees on unconditional untraceability, even when the adversary is able to partially hijack communications and when there are Byzantine participants [108]. In a similar vein, Golle and Juels propose a DC-net construction that is robust to Byzantine participants while retaining the attractive non-interactive property of the original DC-net [53].

Research on DC-nets also moved beyond the construction of primitives to actual system-building. For example, Herbivore is a system that uses DC-nets for anonymous point-to-point communication [50]. DC-nets are designed for sender untraceability, but Herbivore harnesses the fact that by virtue of being a broadcast tool, they also provide receiver untraceability. Therefore, it divides the population into small anonymizing cliques. When Alice wants to send a message to Bob, she first broadcasts the message to her anonymizing clique, which then gets routed to Bob's anonymizing clique. Herbivore managed to scale by reducing the size of each DC-net; however, this limits the anonymity that can be achieved. Another anonymous broadcast system build atop DC-nets is Dissent [28, 112]. Dissent provides stronger accountability against misbehaving par-

ties by preventing a malicious party from flooding the network with spam messages or preventing legitimate nodes from transmitting. It also considers the case where the communication load is asymmetric; for instance, one party wishes to transmit a large file, while the other participants do not wish to transmit anything. The strength and relative scalability of Dissent stems from its use of a shuffle protocol, which guarantees each party one transmit slot during each round of operation. This prevents malicious parties from corrupting other parties' transmissions. However, the original Dissent project scaled to 40 nodes, and even the follow-up work on improving scalability ran on 600 nodes.

We aim to build robust algorithms that can scale to millions of nodes. While DC-nets provide strong, provable anonymity guarantees, current constructions do not yet scale to the numbers of users seen in popular social networks, which motivates our completely different approach. Another difference from DC-nets is that we consider a different adversarial model. Whereas DC-nets provide privacy against worst-case coalitions of nodes in the network, we consider a weaker adversary that can either corrupt a constant fraction of nodes, or observe message's spread at a fixed, unknown point in time. These relaxations of the adversarial model allow us to provide strong privacy guarantees in a distributed, robust fashion.

Other approaches. In recent years, a number of alternative approaches to anonymous broadcast messaging have arisen. One of these is Riposte [27]. Riposte uses a clever adaptation of multi-server PIR [23] to enable a user to anonymously write messages to a bulletin board. Another closely-related approach is to use an anonymous point-to-point communication like Tor [35] to send messages to a public message board. These approaches are scalable (especially when compared to DC-nets), but bulletin boards may be susceptible to spam, since there is no inherent filtering of content.

In the industrial sphere, several anonymous messaging apps have emerged recently, including the now-defunct Secret [2], Whisper [3], and Yik Yak [4]. These applications, which are growing in popularity [26], allow a user to post a microblogging message. The service then spreads the message to the user's contacts without displaying any authorship metadata. However, these services provide no real anonymity guarantees, because authorship metadata is stored on a centralized server, which may be visible to nosy employees, the government, or hackers. Indeed, there is evidence of such information being passed to third parties by Whisper [67].

Unlike approaches for anonymously posting to a bulletin board, we assume messages can only spread over an underlying connectivity graph, such as a social graph. This architecture mimics existing anonymous messaging systems like Secret, Whisper, and Yik Yak, which use connectivity graphs to provide social filtering. However, unlike these industrial solutions, we aim to provide provable anonymity guarantees, even against the service provider itself; we do this by adopting a distributed architecture and developing new algorithms that prevent a strong adversary from learning authorship metadata from a limited view of network traffic. Thus, our work represents a significant departure from existing work, both in terms of problem formulation and technical approach.

Within the realm of statistical message spreading models, the problem of detecting the origin of an epidemic or the source of a rumor has been studied under the *diffusion* model. Recent advances

in [94, 93, 109, 89, 46, 71, 118, 75, 77, 74, 76] show that it is possible to identify the source within a few hops with high probability. Consider an adversary who has access to the underlying *contact network* of friendship links and the snapshot of infected nodes at a certain time. The problem of locating a rumor source, first posed in [94], naturally corresponds to graph-centrality-based inference algorithms: for a continuous time model, [94, 93] used the rumor centrality measure to correctly identify the source after time T (with probability converging to a positive number for large d-regular and random trees, and with probability proportional to $1/\sqrt{T}$ for lines). The probability of identifying the source increases even further when multiple infections from the same source are observed [109]. With multiple sources of infections, spectral methods have been proposed for estimating the number of sources and the set of source nodes in [89, 46]. When infected nodes are allowed to recover as in the susceptible-infected-recovered (SIR) model, Jordan centrality was proposed in [71, 118] to estimate the source. In [118], it is shown that the Jordan center is still within a bounded hop distance from the true source with high probability, independent of the number of infected nodes.

When the adversary collects timestamps (and other metadata) from spy nodes, standard diffusion reveals the location of the source [86, 118, 68]. However, ML estimation is known to be NP-hard [117], and analyzing the probability of detection is also challenging. Nonetheless, even with suboptimal estimators, the source can be effectively identified [86, 118].

Under natural and diffusion-based message spreading – as seen in almost every content-sharing platform today – an adversary with some side-information can identify the rumor source with high confidence. We overcome this vulnerability by asking the reverse question: can we design messaging protocols that spread fast while protecting the anonymity of the source? Related challenges include (a) identifying the best algorithm that the adversary might use to infer the location of the source; (b) providing analytical guarantees for the proposed spreading model; and (c) identifying the fundamental limit on what any spreading model can achieve. We address all of these challenges for snapshot adversarial model (Section 2.2), spy-based adversarial model (Section 2.3), and finally the spy+snapshot model (Section 2.4).

Our work fits into a larger ecosystem that enables anonymous messaging; we implicitly assume that the ecosystem is healthy. For instance, we assume that nodes communicate securely in a distributed fashion, but anonymity-preserving, peer-to-peer (P2P) address lookup is still an active research area [19], as is privacy-preserving distributed data storage in P2P systems [60]. We do not consider adversaries that operate below the application layer (e.g., by monitoring the network or even physical layer) [111, 99]. Lower-level solutions may be more appropriate against such an opponent, harnessing factors like physical proximity of users [1]. In that space, physical layer security and privacy attacks pose a very real threat, as has been documented extensively in prior work [8, 30, 62].

Organization

The remainder of this chapter is organized as follows: To begin, we introduce the general adaptive diffusion protocol in Section 2.1. In Section 2.2, we describe how to specialize adaptive diffusion under a snapshot adversarial model. In Section 2.3, we describe how to apply adaptive diffusion

under a spy-based adversarial model. Combining the key insights of these two approaches, we introduce results from the spy+snapshot adversarial model in Section 2.4. For each adversarial model, we first describe the precise version of adaptive diffusion that applies to infinite d-regular trees, and prove that it achieves perfect obfuscation of the source. We then provide extensions to irregular trees. We conclude by presenting simulated results over real graphs: finite, irregular, and containing cycles. In Section 2.5, we make a connection between the adaptive diffusion on a line and the Pólya's urn process. This connection, while interesting in itself, provides a novel analysis technique for precisely capturing the price of control packets that are passed along with the messages in order to coordinate the spread of messages as per adaptive diffusion. Finally, in Section 2.6, we reiterate the main conclusions to be drawn from this chapter.

2.1 Adaptive diffusion

In this section, we describe adaptive diffusion in its most general form, and leave for later sections the specific choice of parameters involved. For the purpose of introducing adaptive diffusion, we specifically on an infinite d-regular tree as the underlying contact network.

We step through the intuition of the adaptive diffusion spreading model with an example, partially illustrated in Figure 2.2. The precise algorithm description is provided in Protocol 1. Adaptive diffusion ensures that the infected subgraph G_t at any even timestep $t \in \{2, 4, \ldots\}$ is a balanced tree of depth $t/2$, i.e. the hop distance from any leaf to the root (or the center of the graph) is $t/2$. We call the root node of G_t the "virtual source" at time t, and denote it by v_t. We use $v_0 = v^*$ to denote the true source. To keep the regular structure at even timesteps, we use the odd timesteps to transition from one regular subtree G_t to another one G_{t+2} with depth incremented by one.

More concretely, the first three steps are always the same. At time $t = 0$, the rumor source v^* selects, uniformly at random, one of its neighbors to be the virtual source v_2; at time $t = 1$, v^* passes the message to v_2. Next at $t = 2$, the new virtual source v_2 infects all its uninfected neighbors forming G_2 (see Figure 2.2). Then node v_2 chooses to either keep the virtual source token or to pass it along.

If v_2 chooses to remain the virtual source i.e., $v_4 = v_2$, it passes 'infection messages' to all the leaf nodes in the infected subtree, telling each leaf to infect all its uninfected neighbors. Since the virtual source is not connected to the leaf nodes in the infected subtree, these infection messages get relayed by the interior nodes of the subtree. This leads to N_t messages getting passed in total (we assume this happens instantaneously). These messages cause the rumor to spread symmetrically in all directions at $t = 3$. At $t = 4$, no spreading occurs (Figure 2.2, right panel).

If v_2 does not choose to remain the virtual source, it passes the virtual source token to a randomly chosen neighbor v_4, excluding the previous virtual source (in this example, v_0). Thus, if the virtual source moves, it moves away from the true source by one hop. Once v_4 receives the virtual source token, it sends out infection messages. However, these messages do not get passed back in the direction of the previous virtual source. This causes the infection to spread asymmetrically over only one subtree of the infected graph (G_3 in Figure 2.2, left panel). In the subsequent timestep ($t = 4$), the virtual source remains fixed and passes the same infection messages again.

After this second round of asymmetric spreading, the infected graph is once again symmetric about the virtual source v_4 (G_4 in Figure 2.2, left panel).

Adaptive diffusion uses significant amounts of control information to coordinate the spread. In some adversarial models (snapshot), this control information does not hurt anonymity; in others (spy-based), it can be problematic. We therefore introduce different implementations of adaptive diffusion as needed, using different amounts of control information.

In any implementation, the resulting distribution of the random infection process is the same (if the same parameters $\alpha_d(t,h)$ are used). This random infection process can be defined as a time-inhomogeneous (time-dependent) Markov chain over the state defined by the location of the current virtual source $\{v_t\}_{t\in\{0,2,4,\ldots\}}$. By the symmetry of the underlying contact network (which we assume is an infinite d-regular tree) and the fact that the next virtual source is chosen uniformly at random among the neighbors of the current virtual source, it is sufficient to consider a Markov chain over the hop distance between the true source v^* and v_t, the virtual source at time t. Therefore, we design a Markov chain over the state

$$h_t \;=\; \delta_H(v^*, v_t)\;,$$

for even t. Figure 2.2 shows an example with $(h_2, h_4) = (1, 2)$ on the left and $(h_2, h_4) = (1, 1)$ on the right.

At every even timestep, the protocol randomly determines whether to keep the virtual source token ($h_{t+2} = h_t$) or to pass it ($h_{t+2} = h_t + 1$). We specify the resulting time-inhomogeneous Markov chain over $\{h_t\}_{t\in\{2,4,6,\ldots\}}$ by choosing appropriate transition probabilities as a function of time t and current state h_t. For even t, we denote this probability by

$$\alpha_d(t,h) \;\triangleq\; \mathbb{P}\big(\, h_{t+2} = h_t | h_t = h \,\big)\;, \tag{2.1}$$

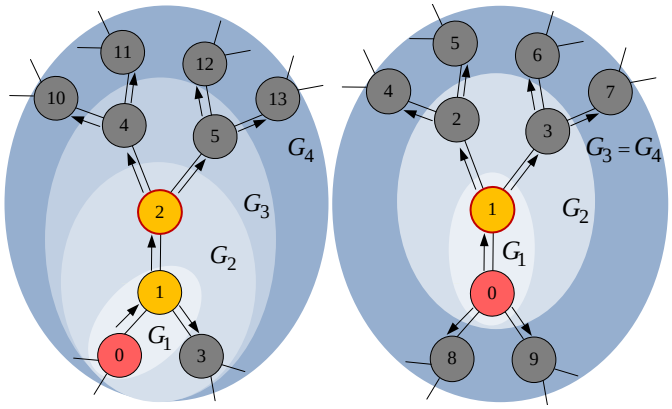


Figure 2.2: Adaptive diffusion over regular trees. Yellow nodes indicate the set of virtual sources (past and present), and for $T = 4$, the virtual source node is outlined in red.

Algorithm 1 Adaptive Diffusion

Input: contact network $G = (V, E)$, source v^*, time T, degree d
Output: set of infected nodes V_T

$V_0 \leftarrow \{v^*\}$, $h \leftarrow 0$, $v_0 \leftarrow v^*$
v^* selects one of its neighbors u at random
$V_1 \leftarrow V_0 \cup \{u\}$, $h \leftarrow 1$, $v_1 \leftarrow u$
let $N(u)$ represent u's neighbors
$V_2 \leftarrow V_1 \cup N(u) \setminus \{v^*\}$, $v_2 \leftarrow v_1$
$t \leftarrow 3$
for $t \leq T$ **do**
    v_{t-1} selects a random variable $X \sim U(0, 1)$
    if $X \leq \alpha_d(t-1, h)$ **then**
        for all $v \in N(v_{t-1})$ **do**
            Infection Message(G,v_{t-1},v,G_t)
    else
        v_{t-1} randomly selects $u \in N(v_{t-1}) \setminus \{v_{t-2}\}$
        $h \leftarrow h + 1$
        $v_t \leftarrow u$
        for all $v \in N(v_t) \setminus \{v_{t-1}\}$ **do**
            Infection Message(G,v_t,v,V_t)
            if $t + 1 > T$ **then**
                break
            Infection Message(G,v_t,v,V_t)
    $t \leftarrow t + 2$
procedure INFECTION MESSAGE(G,u,v,V_t)
    if $v \in V_t$ **then**
        for all $w \in N(v) \setminus \{u\}$ **do**
            Infection Message(G,v,w,G_t)
    else
        $V_t \leftarrow V_{t-2} \cup \{v\}$

where the subscript d denotes the degree of the underlying contact network. In Figure 2.2, at $t = 2$, the virtual source remains at the current node (right) with probability $\alpha_3(2, 1)$, or passes the virtual source to a neighbor with probability $1 - \alpha_3(2, 1)$ (left). The parameters $\alpha_d(t, h)$ fully describe the transition probability of the Markov chain defined over $h_t \in \{1, 2, \ldots, t/2\}$. For example, if we choose $\alpha_d(t, h) = 1$ for all t and h, then the virtual source never moves for $t > 1$. The message spreads almost symmetrically, so the source can be caught with high probability, much like diffusion. If we instead choose $\alpha_d(t, h) = 0$ for all t and h, the virtual source always moves. This ensures that the source is always at one of the leaves of the infected subgraph. We return to this special case when addressing spy-based adversaries in Section 2.3.

The real challenge, then, is choosing the parameters $\alpha_d(t,h)$, which fully specify the virtual source transition probabilities. These parameters can significantly alter the anonymity and spreading properties of adaptive diffusion. In this work, we explain how to choose this parameter α_d to achieve desired source obfuscation.

2.2 Snapshot-based adversarial model

Under the snapshot adversarial model, an adversary observes the infected subgraph G_T at a certain time T and produces an estimate $\hat{v}$ of the source v^* of the message. Since the adversary is assumed to not have any prior information on which node is likely to be the source, we analyze the performance of the maximum likelihood estimator

$$\hat{v}_{ML} = \arg\max_{v \in G_T} \mathbb{P}(G_T|v). \tag{2.2}$$

We show that adaptive diffusion with appropriate parameters can achieve *p*erfect obfuscation, i.e. the probability of detection for the ML estimator when n nodes are infected is close to $1/n$:

$$\mathbb{P}\big(\,\hat{v}_{ML} = v^* | N_T = n\,\big) \;\; = \;\; \frac{1}{n} + o\Big(\frac{1}{n}\Big)\,. \tag{2.3}$$

This is the best source obfuscation that can be achieved by any protocol, since there are only n candidates for the source and they are all equally likely.

Main Result (Snapshot Model)

In this section, we show that for appropriate choice of parameters $\alpha_d(t,h)$, we can achieve both fast spreading and perfect obfuscation over d-regular trees. We start by giving baseline spreading rates for deterministic spreading and diffusion.

Given a contact network of an infinite d-regular tree, $d > 2$, consider the following deterministic spreading protocol. At time $t = 1$, the source node infects all its neighbors. At $t \geq 2$, the nodes at the boundary of the infection spread the message to their uninfected neighbors. Thus, the message spreads one hop in every direction at each timestep. This approach is the fastest-possible spreading, infecting $N_T = 1 + d((d-1)^T - 1)/(d-2)$ nodes at time T, but the source is trivially identified as the center of the infected subtree. In this case, the infected subtree is a balanced regular tree where all leaves are at equal depth from the source.

Now consider a random diffusion model. At each timestep, each uninfected neighbor of an infected node is independently infected with probability q. In this case, $\mathbb{E}[N_T] = 1 + qd((d-1)^T - 1)/(d-2)$, and it was shown in [94] that the probability of correct detection for the maximum likelihood estimator of the rumor source is $\mathbb{P}(\hat{v}_{ML} = v^*) \geq C_d$ for some positive constant C_d that only depends on the degree d. Hence, the source is only hidden in a constant number of nodes close to the center, even when the total number of infected nodes is arbitrarily large.

Now we consider the spreading and anonymity properties of adaptive diffusion. Let $p^{(t)} = [p_h^{(t)}]_{h \in \{1,\ldots,t/2\}}$ denote the distribution of the state of the Markov chain at time t, i.e. $p_h^{(t)} = \mathbb{P}(h_t =$

h). The state transition can be represented as the following $((t/2)+1)\times(t/2)$ dimensional column stochastic matrices:

$$p^{(t+2)} = \begin{bmatrix} \alpha_d(t,1) & & & \\ 1-\alpha_d(t,1) & \alpha_d(t,2) & & \\ & 1-\alpha_d(t,2) & \ddots & \\ & & \ddots & \alpha_d(t,t/2) \\ & & & 1-\alpha_d(t,t/2) \end{bmatrix} p^{(t)}$$

We treat h_t as strictly positive, because at time $t=0$, when $h_0=0$, the virtual source is always passed. Thus, $h_t \geq 1$ afterwards. At all even t, we desire $p^{(t)}$ to be

$$p^{(t)} = \frac{d-2}{(d-1)^{t/2}-1} \begin{bmatrix} 1 \\ (d-1) \\ \vdots \\ (d-1)^{t/2-1} \end{bmatrix} \in \mathbb{R}^{t/2}\,, \tag{2.4}$$

for $d>2$ and for $d=2$, $p^{(t)} = (2/t)\mathbf{1}_{t/2}$ where $\mathbf{1}_{t/2}$ is all ones vector in $\mathbb{R}^{t/2}$. There are $d(d-1)^{h-1}$ nodes at distance h from the virtual source, and by symmetry all of them are equally likely to have been the source:

$$\begin{aligned} \mathbb{P}(G_T|v^*, \delta_H(v^*,v_t)=h) &= \frac{1}{d(d-1)^{h-1}} p_h^{(t)} \\ &= \frac{d-2}{d((d-1)^{t/2}-1)}\,, \end{aligned}$$

for $d>2$, which is independent of h. Hence, all the infected nodes (except for the virtual source) are equally likely to have been the source of the origin. This statement is made precise in Equation (2.7).

Together with the desired probability distribution in Equation (2.4), this gives a recursion over t and h for computing the appropriate $\alpha_d(t,h)$'s. After some algebra and an initial state $p^{(2)}=1$, we get that the following choice ensures the desired Equation (2.4):

$$\alpha_d(t,h) = \begin{cases} \frac{(d-1)^{t/2-h+1}-1}{(d-1)^{t/2+1}-1} & \text{if } d>2 \\ \frac{t-2h+2}{t+2} & \text{if } d=2 \end{cases} \tag{2.5}$$

With this choice of parameters, we show that adaptive diffusion spreads fast, infecting $N_t = O((d-1)^{t/2})$ nodes at time t and each of the nodes except for the virtual source is equally likely to have been the source.

Theorem 2.2.1. *Suppose the contact network is a d-regular tree with $d\geq 2$, and one node v^* in G starts to spread a message according to Protocol 1 at time $t=0$, with $\alpha_d(t,h)$ chosen according to Equation 2.5. At a certain time $T\geq 0$ an adversary estimates the location of the source v^* using the maximum likelihood estimator $\hat{v}_{ML}$. The following properties hold for Protocol 1:*

(a) *the number of infected nodes at time T is*

$$N_T \geq \begin{cases} \frac{2(d-1)^{(T+1)/2}-d}{(d-2)} + 1 & \text{if } d > 2 \\ T+1 & \text{if } d = 2 \end{cases} \tag{2.6}$$

(b) *the probability of source detection for the maximum likelihood estimator at time T is*

$$\mathbb{P}\left(\hat{v}_{ML} = v^*\right) \leq \begin{cases} \frac{d-2}{2(d-1)^{(T+1)/2}-d} & \text{if } d > 2 \\ (1/T) & \text{if } d = 2 \end{cases} \tag{2.7}$$

(c) *the expected hop-distance between the true source v^* and its estimate $\hat{v}_{ML}$ under maximum likelihood estimation is lower bounded by*

$$\mathbb{E}[d(\hat{v}_{ML}, v^*)] \geq \frac{d-1}{d}\frac{T}{2}. \tag{2.8}$$

(Proof in Section A)

Although this choice of parameters achieves perfect obfuscation, the spreading rate is slower than the deterministic spreading model, which infects $O((d-1)^T)$ nodes at time T. However, this type of constant-factor loss in the spreading rate is inevitable: the only way to deviate from the deterministic spreading model is to introduce appropriate delays.

In order to spread according to adaptive diffusion with the prescribed $\alpha_d(h,t)$, the system needs to know the degree d of the underlying contact network. However, performance is insensitive to knowledge of d for certain parameter settings, as shown in the following proposition. Specifically, one can choose $\alpha_d(h,t) = 0$ for all d, h, and t and still achieve performance comparable to the optimal choice. The main idea is that there are as many nodes in the boundary of the snapshot (leaf nodes) as there are in the interior, so it is sufficient to hide among the leaves. One caveat is that if the underlying contact network is a line (i.e. $d = 2$) then this approach fails since there are only two leaf nodes at any given time, and the probability of detection is trivially $1/2$.

Proposition 2.2.1. *Suppose that the underlying contact network G is an infinite d-regular tree with $d > 2$, and one node v^* in G starts to spread a message at time $t = 0$ according to Protocol 1 with $\alpha_d(h,t) = 0$ for all d, h, and t. At a certain time $T \geq 1$ an adversary estimates the location of the source v^* using the maximum likelihood estimator $\hat{v}_{ML}$. Then the following properties hold for the Tree Protocol:*

(a) *the number of infected nodes at time $T \geq 1$ is at least*

$$N_T \;\geq\; \frac{(d-1)^{(T+1)/2}}{d-2}\;; \tag{2.9}$$

(b) *the probability of source detection for the maximum likelihood estimator at time T is*

$$\mathbb{P}\big(\hat{v}_{ML} = v^*\big) \;=\; \frac{d-1}{2+(d-2)N_T}\;;\; \textit{and} \tag{2.10}$$

(c) *the expected hop-distance between the true source v^* and its estimate $\hat{v}$ is lower bounded by*

$$\mathbb{E}[\delta_H(v^*, \hat{v}_{ML})] \;\geq\; \frac{T}{2}\,. \tag{2.11}$$

(Proof in Section A).

Irregular Trees

In this section, we study adaptive diffusion on irregular trees, with potentially different degrees at the vertices. Although the degrees are irregular, we still apply adaptive diffusion with $\alpha_{d_0}(t,h)$'s chosen for a specific d_0 that might be mismatched with the graph due to degree irregularities. There are a few challenges in this degree-mismatched adaptive diffusion. First, finding the maximum likelihood estimate of the source is not immediate, due to degree irregularities. Second, it is not clear a priori which choice of d_0 is good. We first show an efficient message-passing algorithm for computing the maximum likelihood source estimate. Using this estimate, we illustrate through simulations how adaptive diffusion performs and show that the detection probability is not too sensitive to the choice of d_0 as long as d_0 is above a threshold that depends on the degree distribution.

Efficient ML estimation. To keep the discussion simple, we assume that T is even. The same approach can be naturally extended to odd T. Since the spreading pattern in adaptive diffusion is entirely deterministic given the sequence of virtual sources at each timestep, computing the likelihood $\mathbb{P}(G_T|v^*=v)$ is equivalent to computing the probability of the virtual source moving from v to v_T over T timesteps. On trees, there is only one path from v to v_T and since we do not allow the virtual source to "backtrack", we only need to compute the probability of every virtual source sequence $(v_0, v_2, \ldots, v_T)$ that meets the constraint $v_0 = v$. Due to the Markov property exhibited by adaptive diffusion, we have $\mathbb{P}(G_T|\{(v_t,h_t)\}_{t\in\{2,4,\ldots,T\}}) = \prod_{\substack{t<T-1\\ t \text{ even}}} \mathbb{P}(v_{t+2}|v_t,h_t)$, where $h_t = \delta_H(v_0,v_t)$. For t even, $\mathbb{P}(v_{t+2}|v_t,h_t) = \alpha_d(t,h_t)$ if $v_t = v_{t+2}$ and $\frac{1-\alpha_d(t,h_t)}{d_{v_t}-1}$ otherwise. Here d_{v_t} denotes the degree of node v_t in G. Given a virtual source trajectory $\mathcal{P} = (v_0, v_2, \ldots, v_T)$, let $\mathcal{J}_\mathcal{P} = (j_1, \ldots, j_{\delta_H(v_0,v_T)})$ denote the timesteps at which a new virtual source is introduced, with $1 \leq j_i \leq T$. It always holds that $j_1 = 2$ because after $t=0$, the true source chooses a new virtual source and $v_2 \neq v_0$. If the virtual source at $t=2$ were to keep the token exactly once after receiving it (so $v_2 = v_4$), then $j_2 = 6$, and so forth. To find the likelihood of a node being the true source, we sum over all such trajectories

$$\mathbb{P}(G_T|v_0) = \sum_{\mathcal{J}_\mathcal{P}:\mathcal{P}\in\mathcal{S}(v_0,v_T,T)} \underbrace{\frac{1}{d_{v_0}} \prod_{k=1}^{\delta_H(v_0,v_T)-1} \frac{1}{d_{v_{j_k}}-1}}_{A_{v_0}} \times$$
$$\underbrace{\prod_{\substack{t<T\\ t \text{ even}}} \left(\mathbb{1}_{\{t+2\notin\mathcal{J}_\mathcal{P}\}}\alpha_d(t,h_t) + \mathbb{1}_{\{t+2\in\mathcal{J}_\mathcal{P}\}}(1-\alpha_d(t,h_t))\right)}_{B_{v_0}}, \tag{2.12}$$

where $\mathbb{1}$ is the indicator function and $S(v_0, v_T, T) = \{\mathcal{P} : \mathcal{P} = (v_0, v_2, \ldots, v_T)$ is a valid trajectory of the virtual source$\}$. Intuitively, part A_{v_0} of the above expression is the probability of choosing the set of virtual sources specified by $\mathcal{P}$, and part B_{v_0} is the probability of keeping or passing the virtual source token at the specified timesteps. Equation (2.12) holds for both regular and irregular trees. Since the path between two nodes in a tree is unique, and part A_{v_0} is (approximately) the product of node degrees in that path, A_{v_0} is identical for all trajectories $\mathcal{P}$. Pulling A_{v_0} out of the summation, we wish to compute the summation over all valid paths $\mathcal{P}$ of part B_{v_0} (for ease of exposition, we will use B_{v_0} to refer to this whole summation). Although there are combinatorially many valid paths, we can simplify the formula in Equation (2.12) for the particular choice of $\alpha_d(t, h)$'s defined in (2.5).

Proposition 2.2.2. *Suppose that the underlying contact network $\tilde{G}$ is an infinite tree with degree of each node larger than one. One node $\tilde{v}^*$ in $\tilde{G}$ starts to spread a message at time $t = 0$ according to Protocol 1 with the choice of $d = d_0$. At a certain even time $T \geq 0$, the maximum likelihood estimate of $\tilde{v}^*$ given a snapshot of the infected subtree $\tilde{G}_T$ is*

$$\arg \max_{v \in \tilde{G}_T \setminus \tilde{v}_T} \quad \frac{d_0}{d_v} \prod_{v' \in P(\tilde{v}_T, v) \setminus \{\tilde{v}_T, v\}} \frac{d_0 - 1}{d_{v'} - 1} \tag{2.13}$$

where $\tilde{v}_T$ is the (Jordan) center of the infected subtree $\tilde{G}_T$, $P(\tilde{v}_T, v)$ is the unique shortest path from $\tilde{v}_T$ to v, and $d_{v'}$ is the degree of node v'.

To understand this proposition, consider Figure 2.3, which was spread using adaptive diffusion (Protocol 1) with a choice of $d_0 = 2$. Then Equation (2.13) can be computed easily for each node, giving $[1/2, 1, 0, 1, 2/3, 1/2, 1/2, 1/4]$ for nodes $[1, 2, 3, 4, 5, 6, 7, 8]$, respectively. Hence, nodes 2 and 4 are most likely. Intuitively, nodes whose path to the center have small degrees are more likely. However, if we repeat this estimation assuming $d_0 = 4$, then Equation (2.13) gives $[3, 2, 0, 2, 4/3, 3, 3, 3/2]$. In this case, nodes 1, 6, and 7 are most likely. When d_0 is large, adaptive diffusion tends to place the source closer to the leaves of the infected subtree, so leaf nodes are more likely to have been the source.

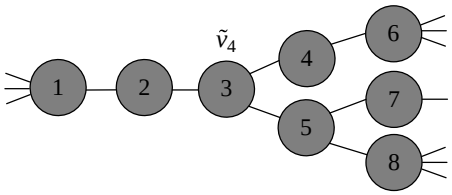


Figure 2.3: Irregular tree $\tilde{G}_4$ with virtual source $\tilde{v}_4$.

Proof of Proposition 2.2.2. We first make two observations: (a) Over regular trees, $\mathbb{P}(G_T|u) = \mathbb{P}(G_T|w)$ for any $u \neq w \in G_T$, even if they are different distances from the virtual souce. (b) Part

B_{v_0} is identical for regular and irregular graphs, as long as the distance from the candidate source node to v_T is the same in both, and the same d_0 is used to compute $\alpha_{d_0}(t,h)$. That is, let $\tilde{G}_T$ denote an infected subtree over an irregular tree network, with virtual source $\tilde{v}_T$, and G_T will denote a regular infected subtree with virtual source v_T. For candidate sources $\tilde{v}_0 \in \tilde{G}_T$ and $v_0 \in G_T$, if $\delta_H(\tilde{v}_T, \tilde{v}_0) = \delta_H(v_T, v_0) = h$, then $B_{v_0} = B_{\tilde{v}_0}$. So to find the likelihood of $\tilde{v}_0 \in \tilde{G}_T$, we can solve for $B_{\tilde{v}_0}$ using the likelihood of $v_0 \in G_T$, and compute $A_{\tilde{v}_0}$ using the degree information of every node in the infected, irregular subgraph.

To solve for $B_{\tilde{v}_0}$, note that over regular graphs, $A_v = 1/(d_0\,(d_0-1)^{\delta_H(v,v_T)-1})$, where d_0 is the degree of the regular graph. If G is a regular tree, Equation (2.12) still applies. Critically, for regular trees, the $\alpha_{d_0}(t,h)$'s are designed such that the likelihood of each node being the true source is equal. Hence,

$$\mathbb{P}(G_T|v_0) = \underbrace{\frac{1}{d_0(d_0-1)^{\delta_H(v_0,v_T)-1}}}_{A_{v_0}} \times B_{v_0}\ , \tag{2.14}$$

is a constant that does not depend on v_0. This gives $B_{v_0} \propto (d_0-1)^{\delta_H(v_T,v_0)}$. From observation (b), we have that $B_{\tilde{v}_0} = B_{v_0}$. Thus we get that for a $\tilde{v}_0 \in \tilde{G}_T \setminus \{\tilde{v}_T\}$,

$$\begin{aligned}\mathbb{P}(\tilde{G}_T|\tilde{v}_0) &= A_{\tilde{v}_0}\, B_{\tilde{v}_0} \\ &\propto \frac{(d_0-1)^{\delta_H(\tilde{v}_T,\tilde{v}_0)}}{d_{\tilde{v}_0}\prod_{\tilde{v}'\in P(\tilde{v}_T,\tilde{v}_0)\setminus\{\tilde{v}_0,\tilde{v}_T\}}(d_{\tilde{v}'}-1)}\end{aligned}$$

After scaling appropriately and noting that $|P(\tilde{v}_T,\tilde{v}_0)| = \delta_H(\tilde{v}_T,\tilde{v}_0)+1$, this gives the formula in Equation (2.13). □

We provide an efficient message passing algorithm for computing the ML estimate in Equation (2.13), which is naturally distributed. We then use this estimator to simulate message spreading for random irregular trees and show that when d_0 exceeds a threshold (determined by the degree distribution), obfuscation is not too sensitive to the choice of d_0.

$A_{\tilde{v}_0}$ can be computed efficiently for irregular graphs with a simple message-passing algorithm. In this algorithm, each node $\tilde{v}$ multiplies its degree information by a cumulative likelihood that gets passed from the virtual source to the leaves. Thus if there are $\tilde{N}_T$ infected nodes in $\tilde{G}_T$, then $A_{\tilde{v}_0}$ for every $\tilde{v}_0 \in \tilde{G}_T$ can be computed by passing $O(\tilde{N}_T)$ messages. This message-passing is outlined in procedure 'Degree Message' of Algorithm 2. For example, consider computing A_5 for the graph in Figure 2.3. The virtual source $\tilde{v}_T = 3$ starts by setting $A_2 = \frac{1}{2}$, $A_4 = \frac{1}{2}$, and $A_5 = \frac{1}{3}$. This gives A_5, but to compute other other values of $A_{\tilde{w}}$, the message passing continues. Each of the nodes $\tilde{v} \in N(3)$ in turn sets $A_{\tilde{w}}$ for their children $\tilde{w} \in N(\tilde{v})$; this is done by dividing $A_{\tilde{v}}$ by $d_{\tilde{w}}$ and replacing the factor of $\frac{1}{d_{\tilde{v}}}$ in $A_{\tilde{v}}$ with $\frac{1}{d_{\tilde{v}}-1}$. For example, node 5 would set $A_7 = \frac{A_5}{2}\cdot\frac{3}{2}$. This step is applied recursively until reaching the leaves.

Algorithm 2 Implementation of ML estimator in (2.13)

Input: infected network $\tilde{G}_T = (\tilde{V}_T, \tilde{E}_T)$, virtual source $\tilde{v}_T$, time T, the spreading model parameter d_0

Output: $\text{argmax}_{\tilde{v}\in\tilde{V}_T}\, \mathbb{P}(\tilde{G}_T|\tilde{v}^* = \tilde{v})$

$P_{\tilde{v}} \triangleq \mathbb{P}(\tilde{G}_T|\tilde{v}^* = \tilde{v})$.
$P_{\tilde{v}_T} \leftarrow 0$
$A_{\tilde{v}} \leftarrow 1$ for $\tilde{v} \in \tilde{V}_T \setminus \{\tilde{v}_T\}$
$A_{\tilde{v}_T} \leftarrow 0$
$A \leftarrow$ Degree Message(G_T, $\tilde{v}_T$, $\tilde{v}_T$, A)
$\mathbb{P}(G_T|v_{leaf}) \leftarrow \frac{1}{d_0(d_0-1)^{T/2-1}} \prod_{\substack{t<T \\ t \text{ even}}} (1 - \alpha_{d_0}(t, \frac{t}{2}))\}$
for all $\tilde{v} \in \tilde{V}_T \setminus \{\tilde{v}_T\}$ **do**
    $h \leftarrow \delta_H(\tilde{v}, \tilde{v}_T)$
    $B_{\tilde{v}} \leftarrow \mathbb{P}(G_T|v_{leaf}) \cdot d_0 \cdot (d_0 - 1)^{h-1}$
    $P_{\tilde{v}} \leftarrow A_{\tilde{v}} \cdot B_{\tilde{v}}$
return $\text{argmax}_{\tilde{v}\in\tilde{V}_T} P_{\tilde{v}}$
procedure DEGREE MESSAGE($\tilde{G}_T$, $\tilde{u}$, $\tilde{v}$, A)
    for all $\tilde{w} \in N(\tilde{v}) \setminus \{\tilde{u}\}$ **do**
        if $\tilde{v} = \tilde{u}$ **then**
            $A_{\tilde{w}} \leftarrow A_{\tilde{v}}/d_{\tilde{w}}$
            Degree Message($\tilde{G}_T$, $\tilde{v}$, $\tilde{w}$, A)
        else
            if $\tilde{v}$ is not a leaf **then**
                $A_{\tilde{w}} \leftarrow A_{\tilde{v}} \cdot d_{\tilde{v}}/(d_{\tilde{w}} \cdot (d_{\tilde{v}} - 1))$
                Degree Message($\tilde{G}_T$, $\tilde{v}$, $\tilde{w}$, A)
    return A

As discussed earlier, $B_{\tilde{v}_0}$ only depends on d_0 and $\delta_H(\tilde{v}_T, \tilde{v}_0)$. If $v_{\text{leaf}} \in G_T$ is a leaf node and G is a regular tree, we get

$$\mathbb{P}(G_T|v_{\text{leaf}}) = \underbrace{\frac{1}{d_0(d_0-1)^{T/2-1}}}_{A_{v_{\text{leaf}}}} \underbrace{\prod_{\substack{t<T \\ t \text{ even}}} (1 - \alpha_{d_0}(t, \frac{t}{2}))}_{B_{v_{\text{leaf}}}}. \tag{2.15}$$

If $\tilde{v}_0$ is $h < T/2$ hops from $\tilde{v}_T$, then for node v_0 with $\delta_H(v_0, v_T) = h < T/2$ over a regular tree,

$$\mathbb{P}(G_T|v_0) = \mathbb{P}(G_T|v_{\text{leaf}}) = \frac{1}{d_0 \cdot (d_0-1)^{h-1}} B_{v_0}.$$

Finally, $B_{\tilde{v}_0} = B_{v_0}$. So to solve for B_5 in our example, we compute $\mathbb{P}(G_T|v_{leaf})$ for a 3-regular graph at time $T = 4$. This gives $\mathbb{P}(G_4|v_{leaf}) = A_{v_{leaf}} \cdot B_{v_{leaf}} = \frac{1}{6} \cdot (1 - \alpha_3(2, 1)) = \frac{1}{9}$. Thus $B_5 = \mathbb{P}(G_4|v_{leaf}) \cdot d_0 \cdot (d_0 - 1)^{h-1} = \mathbb{P}(G_4|v_{leaf}) \cdot 3 \cdot (2)^0 = \frac{1}{3}$. This gives $\mathbb{P}(\tilde{G}_4|5) = A_5 \cdot B_5 = \frac{1}{9}$. The same can be done for other nodes in the graph to find the maximum likelihood source estimate.

Probability of detection. In this section, we provide the probability of detection for adaptive diffusion over trees whose node degrees are drawn i.i.d. from some distribution D. However, we cannot exactly use the ML estimator from Equation 2.13, which assumes the infinite irregular tree G is given, and the source v^* is chosen randomly from the nodes of G. Equation 2.13 is the correct ML estimator in any practical scenario, but analyzing the probability of detection under this model requires a prior on the (infinitely many) nodes of G. We therefore consider a closely-related random process, in which we fix a source v^* and generate G (and consequently, G_T) on-the-fly. Specifically, at time $t = 0$, v^* draws a degree d_{v^*} from D, and generates d_{v^*} child nodes. The source picks one of these neighbors uniformly at random to be the new virtual source. Each time a node v is infected according to Protocol 1, v draws its degree d_v from D, then generates $d_v - 1$ child nodes. For example, as soon as v_2, neighbor of v^*, receives the virtual source token, it draws its degree from D and generates $d_{v_2} - 1$ children. The structure of the underlying, infinite contact network G is independent of G_T conditioned on the uninfected neighbors of the leaves of G_T, and need not be considered. The adversary observes $\mathcal{G}_T$, which is an unlabeled snapshot including G_T and its uninfected neighbors. We have that $\mathbb{P}(\hat{v}_{MAP} = v^*|T) = \sum_{\mathcal{G}_T} \mathbb{P}(\mathcal{G}_T|T)\mathbb{P}(\hat{v}_{MAP} = v^*|\mathcal{G}_T)$. We first consider $\mathbb{P}(\hat{v}_{ML} = v^*|\mathcal{G}_T)$.

Theorem 2.2.2. *Suppose a node v^* starts to spread a message at time $t = 0$ according to Protocol 1 with $\alpha_d(t, h) = 0$ for all t, h. The underlying irregular tree is generated according to the random branching process described above. At a certain even time $T \geq 0$, we compute an estimate of v^* given a snapshot of the infected subtree $\mathcal{G}_T$. The following results hold:*

(a) *The MAP estimator $\hat{v}_{MAP} = \arg\max_v \mathbb{P}(v = v^*|\mathcal{G}_T)$ is*

$$\hat{v}_{MAP} = \arg\min_{v \in \partial G_T} \prod_{\substack{w \in \phi(v, v_T) \\ \setminus \{v_T, v\}}} (d_w - 1) \tag{2.16}$$

where ∂G_T denotes the leaves of G_T, and $\phi(v, v_T)$ denotes the unique shortest path between leaf v and virtual source v_T.

(b) *The probability of detection using the estimator in Equation 2.16 is*

$$\mathbb{P}(\hat{v}_{ML} = v^*|\mathcal{G}_T) = \frac{1}{d_{v_T} \min_{v \in \partial G_T} \prod_{\substack{w \in \phi(v, v_T) \\ \setminus \{v_T, v\}}} (d_w - 1)}. \tag{2.17}$$

(Proof in Section A).

Given a degree distribution D and parameter T, we characterize the expression in Equation (2.17). We use D to denote both a random variable and its distribution—the distinction should be clear from context. The random variable D has support $\boldsymbol{f} = (f_1, \ldots, f_\eta)$, and probabilities $\boldsymbol{p} = (p_1, \ldots, p_\eta)$. As before, we assume each node in the irregular tree has degree at least 2. Without loss of generality, we assume $2 \leq f_1 < \ldots < f_\eta$. We also assume the branching process is not a line graph, so D's support set has at least two elements, i.e., $\eta \geq 2$.

Theorem 2.2.3. *Given a tree G_T as defined previously, we define*

$$\Lambda_{G_T} \equiv d_{v_T} \min_{v \in \partial G_T} \prod_{\substack{w \in \phi(v, v_T) \\ \setminus \{v_T, v\}}} (d_w - 1).$$

The following results hold:

(a) *If $p_1(f_1 - 1) > 1$,*

$$\mathbb{P}\left(\left| \frac{\log(\Lambda_{G_T})}{T/2} - \log(f_1 - 1) \right| > \delta \right) \leq e^{-C_D T} \tag{2.18}$$

for time $T \geq C'_D$, where C_D and C'_D are constants that depend only on the degree distribution and the choice of $\delta > 0$.

(b) *If $p_1(f_1 - 1) \leq 1$, define the mean number of children as*

$$\mu_D \equiv \sum_{i=1}^{\eta} p_i(f_i - 1)$$

and the set

$$\mathcal{R}_D \;=\; \left\{ \boldsymbol{r} \in S_\eta \mid \log(\mu_D) \geq D_{KL}(\boldsymbol{r} \| \boldsymbol{\beta}) \right\}, \tag{2.19}$$

where S_η denotes the η-dimensional probability simplex, $D_{KL}(\cdot\|\cdot)$ denotes KL divergence, and $\boldsymbol{\beta}$ is a length-η vector in which $\beta_i = p_i(f_i - 1)/\mu_D$. Further, define $\boldsymbol{r}^*$ as follows:

$$\boldsymbol{r}^* = \; \arg\min_{\boldsymbol{r} \in \mathcal{R}_D} \; \langle \boldsymbol{r}, \log(\boldsymbol{f} - 1) \rangle \tag{2.20}$$

where $\langle \boldsymbol{r}, \log(\boldsymbol{f} - 1) \rangle = \sum_{i=1}^{\eta} r_i \log(f_i - 1)$. Then for any $\delta > 0$

$$\mathbb{P}\left(\left| \frac{\log(\Lambda_{G_T})}{T/2} - \langle \boldsymbol{r}^*, \boldsymbol{f} \rangle \right| > \delta \right) \;\leq\; e^{-C_{D',\delta} T} \tag{2.21}$$

for all $T \geq C'_{D,\delta}$, where $C_{D,\delta}$ and $C'_{D,\delta}$ are positive constants that depend only on the degree distribution D and the choice of $\delta > 0$.

(Proof in Section A).

This theorem says that the probably of detection concentrates in a way that depends heavily on the minimum degree of the degree distribution. Since the number of total nodes in the tree scales according to the mean of the degree distribution, not the minimum, adaptive diffusion does not achieve perfect hiding over irregular trees.

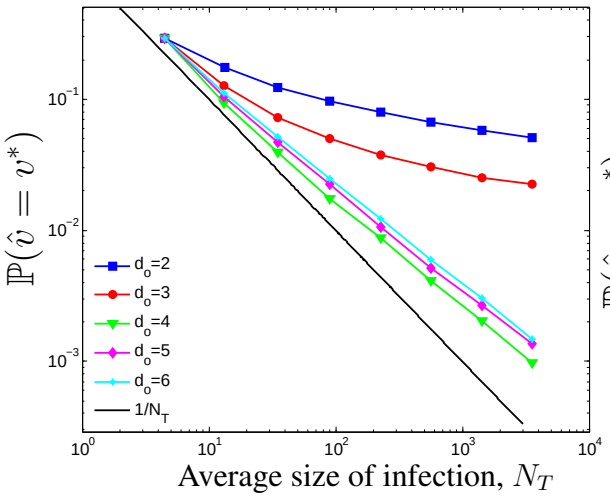


Figure 2.4: The probability of detection by the maximum likelihood estimator depends on the assumed degree d_0; the source cannot hide well below a threshold value of d_0.

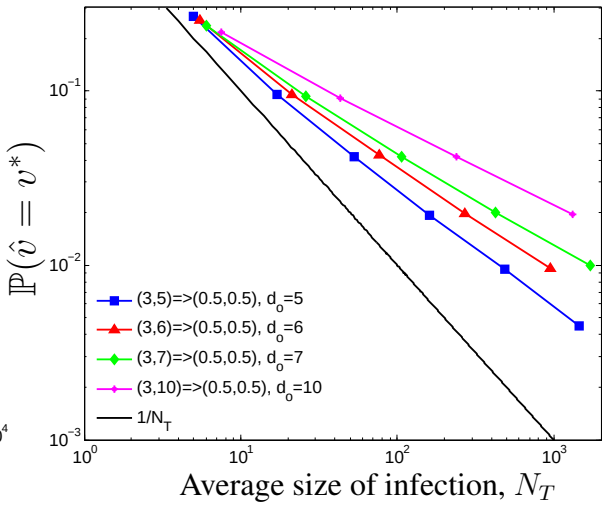


Figure 2.5: Adaptive diffusion no longer provides perfect obfuscation for highly irregular graphs.

Simulation studies

We tested adaptive diffusion over random trees; each node's degree was drawn i.i.d. from a fixed distribution. Figure 2.4 illustrates simulation results for random trees in which each node has degree 3 or 4 with equal probability, averaged over 100,000 trials. The number of nodes infected scales as $N_T \sim \mathbb{E}[D-1]^T = 2.5^T$, where D represents the degree distribution of the underlying random irregular tree. The value of d_0 corresponds to a regular tree with size scaling as $(d_0-1)^T$. Hence, one can expect that for $d_0 - 1 < 2.5$, the source is likely to be in the center of the infection, and for $d_0 > 2.45$ the source is likely to be at the boundary of the infection. Since the number of nodes in the boundary is significantly larger than the number of nodes in the center, the detection probability is lower for $d_0 - 1 > 2.5$. This is illustrated in the figure, which matches our prediction. In general, $d_0 = 1 + \lceil \mathbb{E}[D-1] \rceil$ provides the best obfuscation, and it is robust for any value above that. In this plot, data points represent successive even timesteps; their uniform spacing implies the message is spreading exponentially quickly.

Figure 2.5 illustrates the probability of detection as a function of infection size while varying the degree distribution of the underlying tree. The notation $(3,5) => (0.5,0.5)$ in the legend indicates that each node in the tree has degree 3 or 5, each with probability 0.5. For each distribution tested, we chose d_0 to be the maximum degree of each degree distribution. The average size of infection scales as $N_T \sim \mathbb{E}[D-1]^T$ as expected, whereas the probability of detection scales as $(d_{min}-1)^{-T} = 2^{-T}$, which is independent of the degree distribution. This suggests that adaptive diffusion fails to provide near-perfect obfuscation when the underlying graph is irregular, and the gap increases with the irregularity of the graph.

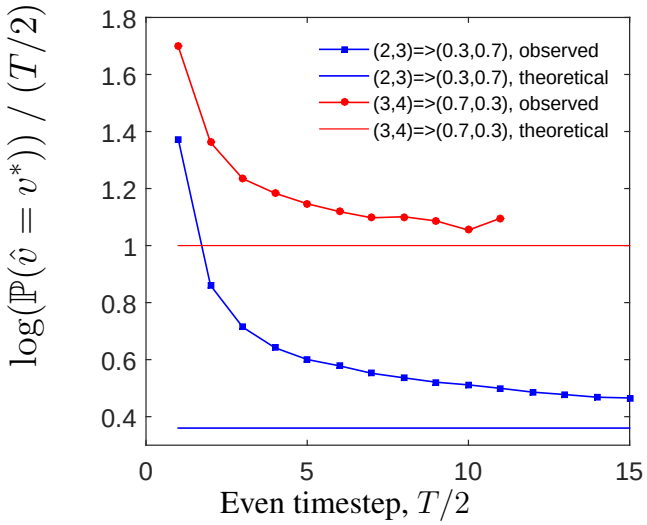


Figure 2.6: Empirical verification of Theorems 2.2.2 and 2.2.3. We observe that the probability of detection converges in time to the predicted values, which depend only on the underlying degree distribution.

Figure 2.6 empirically checks the predictions in Theorems 2.2.2 and 2.2.3. The distribution with support $\boldsymbol{f} = (3, 4)$ with probabilities $\boldsymbol{p} = (0.5, 0.5)$ addresses case 1 from the theorem, where $p_1(f_1 - 1) > 1$. The distribution with support $\boldsymbol{f} = (2, 3)$ with probabilities $\boldsymbol{p} = (0.3, 0.7)$ addresses case 2, where $p_1(f_1 - 1) < 1$. In both examples, we observe that the empirical $\log(\mathbb{P}(\hat{v} = v^*))/(T/2)$ converges to the theoretical value predicted in Figure 2.6. However, this convergence may be slow, and the timestep duration of these experiments was limited by computational considerations since the graph size grows exponentially in time.

General Graphs

In this section, we demonstrate how adaptive diffusion fares over graphs that involve cycles, irregular degrees, and finite graph size. We provide theoretical guarantees for the special case of two-dimensional grid graphs, and we show simulated results over a social graph dataset.

Grid graphs

Here, we derive the optimal parameters $\alpha(t, h)$ for spreading with adaptive diffusion over an infinite grid graph, defined as the graph Cartesian product of two infinite line graphs. This example highlights challenges associated with spreading over cyclic graphs, while still providing a regular, symmetric structure. To spread over grids, we make some changes to the adaptive diffusion protocol, outlined in Protocol 8 (grid adaptive diffusion).

First, standard adaptive diffusion requires the virtual source to know its distance from the true source. Over trees, this information was transmitted by passing a distance counter, h_t, that was incremented each time the virtual source changed; since the network was a tree, this distance from

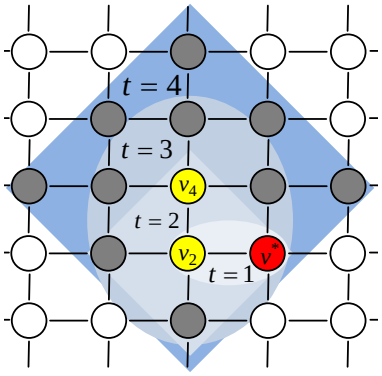


Figure 2.7: Grid adaptive diffusion spreading pattern.

the source was non-decreasing as long as the virtual source was non-backtracking. However, on a cyclic graph (e.g., a grid), the virtual source's non-backtracking random walk could actually cause its distance from the true source to decrease with time. We wish to avoid this to preserve adaptive diffusion's anonymity guarantees.

Therefore, instead of passing the raw hop distance h_t to each new virtual source, grid adaptive diffusion passes directional coordinates (h_t^H, h_t^V) detailing the virtual source's horizontal and vertical displacement from the source, respectively. For example, in Figure 2.7, the virtual source v_4 would receive parameters $(h_t^H, h_t^V) = (-1, 1)$ because it is one hop west and one hop north of the true source. This indexing assumes some notion of directionality over the underlying contact network; nodes should know whether they received a message from the north, south, east, or west. If a virtual source chooses to move, it always passes the token to a node that is further away from the true source, i.e. $|h_{t+1}^H| + |h_{t+1}^V| \geq |h_t^H| + |h_t^V|$.

To maintain symmetry about the virtual source, we also modify the message-passing algorithm. Just as in adaptive diffusion over trees, when a new virtual source sends out branching messages, it sends them in every direction except that of the old virtual source. However, unlike adaptive diffusion over trees, each branch message has up to two "forbidden" directions: the direction of the previous virtual source, and the direction of the node that originated the branching message (these might be the same). Thus, if a branch message is sent west, and the previous virtual source was south of the current virtual source, each node would only propagate the message west and/or north. Whenever a node receives a branch message and its neighbors are not all infected, it infects all uninfected neighbors. As in adaptive diffusion over trees, two waves of directional branching messages are sent each time the virtual source moves, in every direction except that of the old virtual source. If the virtual source instead chooses to stay fixed, then the same rules hold, except the new virtual source only sends one wave of branch messages, symmetrically in every direction.

Given the spreading protocol, we can choose $\alpha(t, h)$ to give optimal hiding:

$$\alpha(t, h) = \frac{t - 2(h-1)}{t+4}. \tag{2.22}$$

Under these conditions, the following result shows that we achieve perfect obfuscation, i.e. $\mathbb{P}(\hat{v}_{ML} = v^*) = 1/N_T + o(1/N_T)$.

Proposition 2.2.3. *Suppose the contact network is an infinite grid, and one node v^* in G starts to spread a message according to Protocol 8 (grid adaptive diffusion) at time $t = 0$, with $\alpha(t, h)$ chosen according to Equation (2.22). At a certain time $T \geq 0$ an adversary estimates the location of the source v^* using the maximum likelihood estimator $\hat{v}_{ML}$. The following properties hold for Protocol 8:*

(a) *the number of infected nodes at time T is*

$$N_T \geq \frac{(T+1)^2}{2} \tag{2.23}$$

(b) *the probability of source detection for the maximum likelihood estimator at time T is*

$$\mathbb{P}\left(\hat{v}_{ML} = v^*\right) \leq \frac{2}{(T+3)(T-1)}. \tag{2.24}$$

(Proof in Section A)

The baseline infection rate for deterministic, symmetric spreading is $N_T = T^2 + (T+1)^2$. Grid adaptive diffusion infection rate is within a constant factor of this maximum possible rate, and it achieves perfect obfuscation over grid graphs. The price to pay for this non-tree graph is that (a) a significant amount of metadata needs to be transmitted to coordinate the spread—particularly with respect to the directionality of messages; and (b) the position of the nodes w.r.t. a global reference needs to be known. Hence, the current implementation of the grid adaptive diffusion has a limited scope, and it remains an open question how to avoid such requirements for grids and still achieve a perfect obfuscation.

Real-world social graphs

In this section, we provide simulation results from running adaptive diffusion over an underlying connectivity network of 10,000 Facebook users, as described by the Facebook WOSN dataset [107]. We eliminated all nodes with fewer than three friends (this approach is taken by several existing anonymous applications so users cannot guess which of their friends originated the message), which left us with a network of 9,502 users.

Over this underlying network, we selected a node uniformly at random as the rumor source, and spread the message using adaptive diffusion for trees. We did not use grid adaptive diffusion because Protocol 8 assumes the underlying graph has a symmetric structure with a global notion of directionality, whereas the tree-based adaptive diffusion makes no such assumptions. We set $d_0 = \infty$, which means that the virtual source is always passed to a new node (i.e., $\alpha_d(t, h) = 0$). This choice is to make the ML source estimation faster; other choices of d_0 may outperform this naive choice. To preserve the symmetry of our constructed trees as much as possible, we constrained

each infected node to infect a maximum of three other nodes in each timestep. We also give the adversary access to the undirected infection *s*ubtree that explicitly identifies all pairs of nodes for which one node spread the infection to the other. This subtree is overlaid on the underlying contact network, which is not necessarily tree-structured. We demonstrate in simulation (Figure 2.8) that even with this strong side information, the adversary can only identify the true message source with low probability.

Using the naive method of enumerating every possible message trajectory, it is computationally expensive to find the exact ML source estimate since there are 2^T possible trajectories, depending on whether the virtual source stayed or moved at each timestep. If the true source is one of the leaves, we can closely approximate the ML estimate among all leaf nodes, using the same procedure as described in 2.2, with one small modification: in graphs with cycles, the term $(d_{v_{j_k}} - 1)$ from equation (2.12) should be substituted with $(d^u_{v_{j_k}} - 1)$, where $d^u_{v_{j_k}}$ denotes the number of uninfected neighbors of v_{j_k} at time j_k. Loops in the graph cause this value to be time-varying, and also dependent on the location of v_0, the candidate source. We did not approximate the ML estimate for non-leaves because the simplifications used in Section 2.2 to compute the likelihood no longer hold, leading to an exponential increase in the problem dimension.

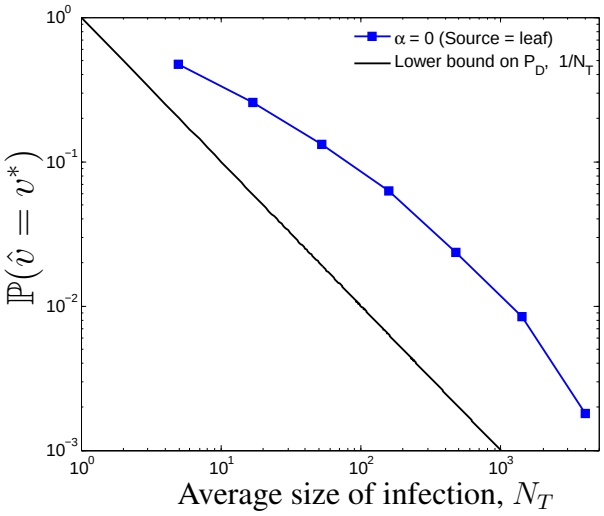


Figure 2.8: Near-ML probability of detection for the Facebook graph with adaptive diffusion.

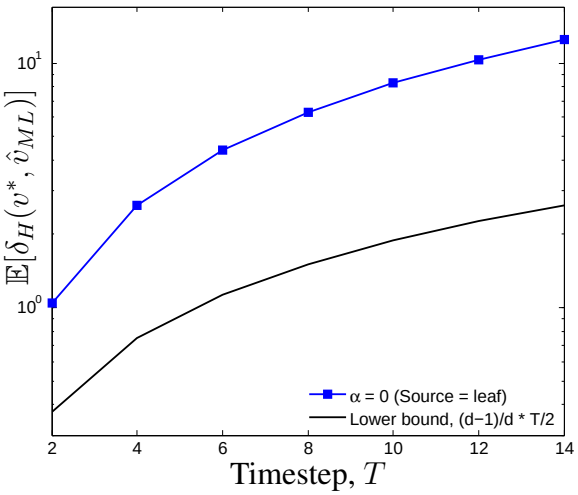


Figure 2.9: Hop distance between true source and estimated source over infection subtree for adaptive diffusion over the Facebook graph.

This approach is only an approximation of the ML estimate because the virtual source could move in a loop over the social graph (i.e., the same node could be the virtual source more than once, in nonadjacent timesteps).

On average, adaptive diffusion reached 96 percent of the network within 10 timesteps using $d_0 = 4$. We also computed the average distance of the true source from the estimated source over the infected subtree (Figure 2.9). We see that as time progresses, so does the hop distance of

the estimated source from the true source. In social networks, nearly everyone is within a small number of hops (say, 6 hops [106]) from everyone else, so this computation is not as informative in this setting. However, it is relevant in location-based connectivity graphs, which can induce large hop distances between nodes.

2.3 Spy-based adversarial model

The spy-based adversary is very different from the snapshot adversary. In this section, we prove that over d-regular trees, choosing $\alpha_d(t, h) = 0$ gives asymptotically optimal hiding in d.

For the spy-based adversary, we model each node other than the source as a spy with probability p. At some point in time, the source node v^* starts propagating its message over the graph according to some spreading protocol (e.g., diffusion or adaptive diffusion). Each spy node $s_i \in V$ observes: (1) the time T_{s_i} (relative to an absolute reference) at which it receives the message, (2) the parent node p_{s_i} that relayed the message, and (3) any other metadata used by the spreading mechanism (such as control signaling in the message header). At some time, spies *a*ggregate their observations; using the collected metadata and the structure of the underlying graph, the adversary estimates the author of the message, $\hat{v}$.

To define perfect obfuscation for this adversarial model, we first observe the following:

Proposition 2.3.1. *Under a spy-based adversary, no spreading protocol can have a probability of detection less than p.*

This results from considering the first-spy estimator, which returns the parent of the first spy to observe the message. Regardless of spreading, this estimator returns the true source with probability at least p; with probability p, the first node (other than the true source) to see the message is a spy.

We therefore say a protocol achieves perfect obfuscation against a spy-based adversary if the ML probability of detection conditioned on the spy probability p is bounded by

$$\mathbb{P}\big(\,\hat{v}_{ML} = v^* | p\,\big) \;\; = \;\; p + o\Big(p\Big)\,. \tag{2.25}$$

For standard diffusion spreading, finding a computationally-efficient algorithm for (near-) optimal maximum likelihood (ML) message source inference is an open problem under the spy-based adversarial model, as is the corresponding detection probability analysis. Recent works [86, 117] have focused on identifying the message source through heuristic, low-cost algorithms. These findings suggest that a spy-based adversary with metadata can locate the source with high probability under diffusion spreading.

Indeed, when the underlying graph is a d-regular tree, the probability of detection increases over time, since the estimator receives more information. Moreover, it is straightforward to show that the probability of detection tends to 1 as degree of the underlying graph $d \to \infty$:

Proposition 2.3.2. *Suppose the contact network is a regular tree with degree d. There is a source node v^*, and each node other than the source is chosen to be a spy node i.i.d. with probability p as*

described in the spy model. In each timestep, each infected node infects each uninfected neighbor independently with probability q. Then the probability of detection $\mathbb{P}(\hat{v}_{ML} = v^*) \geq 1 - (1 - qp)^d$.

This bound implies that as degree increases, the probability of detecting the true source of diffusion approaches 1. The proposition also results from the first-spy estimator. We consider all neighbors of v^* that (a) are spies and (b) receive the message at $t = 1$. If there is at least one such node, then the source is identified with probability 1. Each neighbor of v^* meets these criteria with probability pq.

These observations suggest that diffusion provides poor anonymity guarantees in real networks; contact networks may be high degree, and the adversary is not time-constrained.

Main result (Spy-based adversary)

In this section, we prove that over d-regular trees, adaptive diffusion with $\alpha_d(t, h) = 0$ achieves asymptotically perfect obfuscation in d. We also show that adaptive diffusion hides the source better than diffusion over d-regular trees, $d > 2$. However, to prove this result, we require a modified implementation of adaptive diffusion when $\alpha_d(t, h) = 0$. This implementation, which we call the Tree Protocol, facilitates analysis and is also fully distributed, avoiding the explicit notion of a virtual source.

Tree Protocol. Under adaptive diffusion with $\alpha_d(t, h) = 0$, the goal is to build an infected subtree with the true source at one of the leaves. The Tree Protocol reproduces this spreading pattern by proceeding as follows: Whenever a node v passes a message to node w, it includes three pieces of metadata: (1) the parent node $p_w = v$, (2) a binary direction indicator $u_w \in \{\uparrow, \downarrow\}$, and (3) the node's level in the infected subtree $m_w \in \mathbb{N}$. The parent p_w is the node that relayed the message to w. The direction bit u_w flags whether node w is a *spine* node, responsible for increasing the depth of the infected subtree. The level m_w describes the hop distance from w to the nearest leaf node in the final infected subtree, as $t \to \infty$. The parent metadata is included purely to facilitate the adversary's source estimation. Even with this extra metadata, the tree protocol achieves asymptotically optimal hiding.

At time $t = 0$, the source chooses a neighbor uniformly at random (e.g., node 1) and passes the message and metadata $(p_1 = 0,\ u_1 = \uparrow,\ m_1 = 1)$. Figure 2.10 illustrates an example spread for the non-distributed version of this protocol, which we now adapt for the distributed version in Protocol 3. In this example, node 0 initially passes the message to node 1. Yellow denotes spine nodes, which receive the message with $u_w = \uparrow$, and gray denotes those that receive it with $u_w = \downarrow$. Whenever a node w receives a message , there are two cases. If $u_w = \uparrow$, node w forwards the message to another neighbor z chosen uniformly at random with 'up' metadata: $(p_z = w,\ u_z = \uparrow,\ m_z = m_w + 1)$. All of w's remaining neighbors z' receive the message with 'down' metadata: $(p_{z'} = w,\ u_{z'} = \downarrow,\ m_{z'} = m_w - 1)$. In Figure 2.10, node 1 passes the 'up' message to node 2 and the 'down' message to node 3. On the other hand, if $u_w = \downarrow$ and $m_w > 0$, node w forwards the message to all its remaining neighbors with 'down' metadata: $(p_z = w,\ u_z = \downarrow,\ m_z = m_w - 1)$. If a node receives $m_w = 0$, it does not forward the message further.

As before, this protocol ensures that the infected subgraph is a symmetric tree with the true source at one of the leaves. The key difference between Protocol 1 (naive adaptive diffusion) with $\alpha_d(t, h) = 0$ and Protocol 3 (Tree Protocol) is that the latter does not rely on message-passing from the virtual source to control spreading. Instead, it passes enough control information to realize the same spreading pattern in a fully-distributed fashion.

Protocol 3 Tree Protocol

Input: contact network $G = (V, E)$, source v^*, time T
Output: infected subgraph $G_T = (V_T, E_T)$

$V_0 \leftarrow \{v^*\}$
$m_{v^*} \leftarrow 0$ and $u_{v^*} \leftarrow\uparrow$
v^* selects one of its neighbors w at random
$V_1 \leftarrow V_0 \cup \{w\}$
$m_w \leftarrow 1$ and $u_w \leftarrow\uparrow$
$t \leftarrow 2$
for $t \leq T$ **do**
    for all $v \in V_{t-1}$ with uninfected neighbors and $m_v > 0$ **do**
        if $u_v = \uparrow$ **then**
            v selects one of its uninfected neighbors w at random
            $V_t \leftarrow V_{t-1} \cup \{w\}$
            $m_w \leftarrow m_w + 1$ and $u_w \leftarrow\uparrow$
        for all uninfected neighboring nodes z of v **do**
            $V_t \leftarrow V_{t-1} \cup \{z\}$
            $u_z \leftarrow\downarrow$ and $m_z \leftarrow m_v - 1$
    $t \leftarrow t + 1$

In the spy-based adversarial model, each spy s_i in the network observes any received messages, the associated metadata, and a timestamp T_{s_i}. Figure 2.11 illustrates the information observed by each spy node, where spies are outlined in red.

Source Estimation. The precise ML estimation algorithm is detailed in Algorithm 4. Because adaptive diffusion has deterministic timing, spies only help the estimator discard candidate nodes. We assume the message spreads for infinite time. There is at least one spy on the spine; consider the first such spy to receive the message, s_0. This spine spy (along with its parent and level metadata) allows the estimator to specify a feasible subtree in which the true source must lie. In Figure 2.10, node 8 is on the spine with level $m_8 = 4$, so the feasible subtree is rooted at node 5 and contains all the pictured nodes except node 8 (9's children and grandchildren also belong, but are not pictured). Spies outside the feasible subtree do not influence the estimator, because their information is independent of the source conditioned on s_0's metadata. Only leaves of the feasible subtree could have been the source—e.g., nodes 0, 3, 6, and 7, as well as 9's grandchildren.

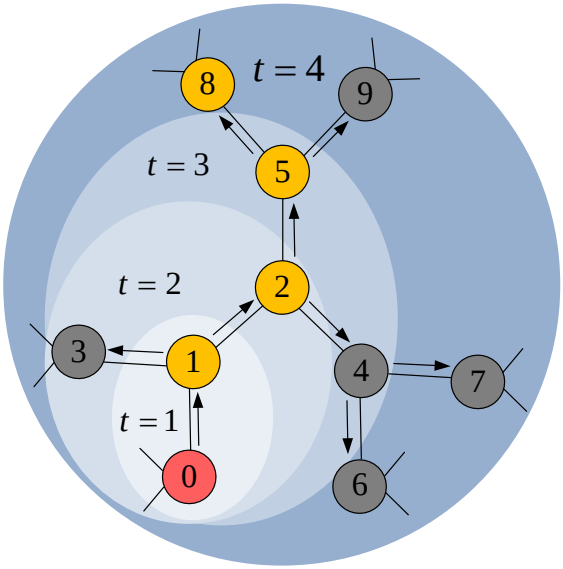


Figure 2.10: Message spread using the tree protocol (Protocol 3).

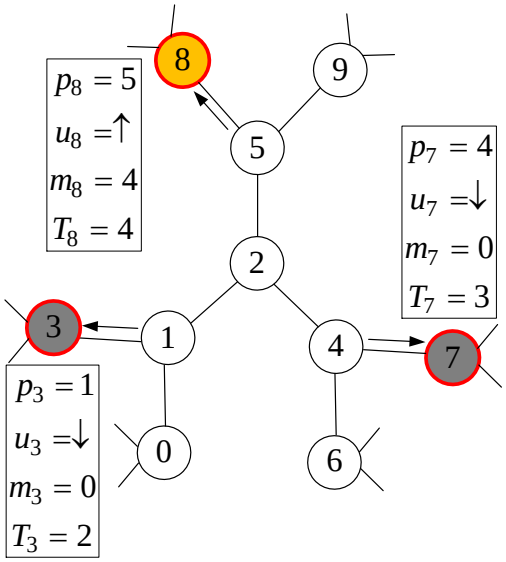


Figure 2.11: The information observed by the spy nodes 3, 7, and 8 for the spread in Figure 2.10. Timestamps in this figure are absolute, but they need not be.

Protocol 4 ML Source Estimator for Algorithm 3

Input: contact network $G = (V, E)$, spy nodes $S = \{s_0, s_1 \ldots\}$ and metadata $s_i : (p_{s_i}, m_{s_i}, u_{s_i})$
Output: ML source estimate $\hat{v}_{ML}$

Let s_0 denote the lowest-level spine spy, with metadata $(p_{s_0}, m_{s_0}, u_{s_0})$.
$\tilde{V} \leftarrow \{v \in V : \delta_H(v, s_0) \leq m_{s_0}$ and $p_{s_0} \in \mathcal{P}(v, s_0)\}$
$\tilde{E} \leftarrow \{(u, v) : (u, v) \in E$ and $u, v \in \tilde{V}\}$
Define the feasible subgraph as $F(\tilde{V}, \tilde{E})$
$L \leftarrow \emptyset$ ▷ Set of feasible pivots
$K \leftarrow \emptyset$ ▷ Set of eliminated pivot neighbors
for all $s \in S$ with $s \in \tilde{V}$ **do**
    Let $\begin{bmatrix} h_{s,\ell_s} \\ h_{\ell_s,s_0} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \cdot \begin{bmatrix} |P(s, s_0)| \\ T_{s_0} - T_s \end{bmatrix}$
    $\ell_s \leftarrow v \in \mathcal{P}(s, s_0) : \delta_H(s, \ell_s) = h_{s,\ell_s}$
    $k_s \leftarrow v \in \mathcal{P}(s, s_0) : \delta_H(s, k_s) = h_{s,\ell_s} - 1$
    $L \leftarrow L \cup \{\ell_s\}$ ▷ Add pivot
    $K \leftarrow K \cup \{k_s\}$ ▷ Add pivot neighbor
Find the lowest-level pivot: $\ell_{min} \leftarrow \text{argmin}_{\ell \in L} m_\ell$
$U \leftarrow \emptyset$ ▷ Candidate sources
for all $v \in \tilde{V}$ where v is a leaf in $F(\tilde{V}, \tilde{E})$ **do**
    if $\mathcal{P}(v, \ell_{min}) \cap K = \emptyset$ **then**
        $U \leftarrow U \cup \{v\}$
return $\hat{v}_{ML}$, drawn uniformly from U

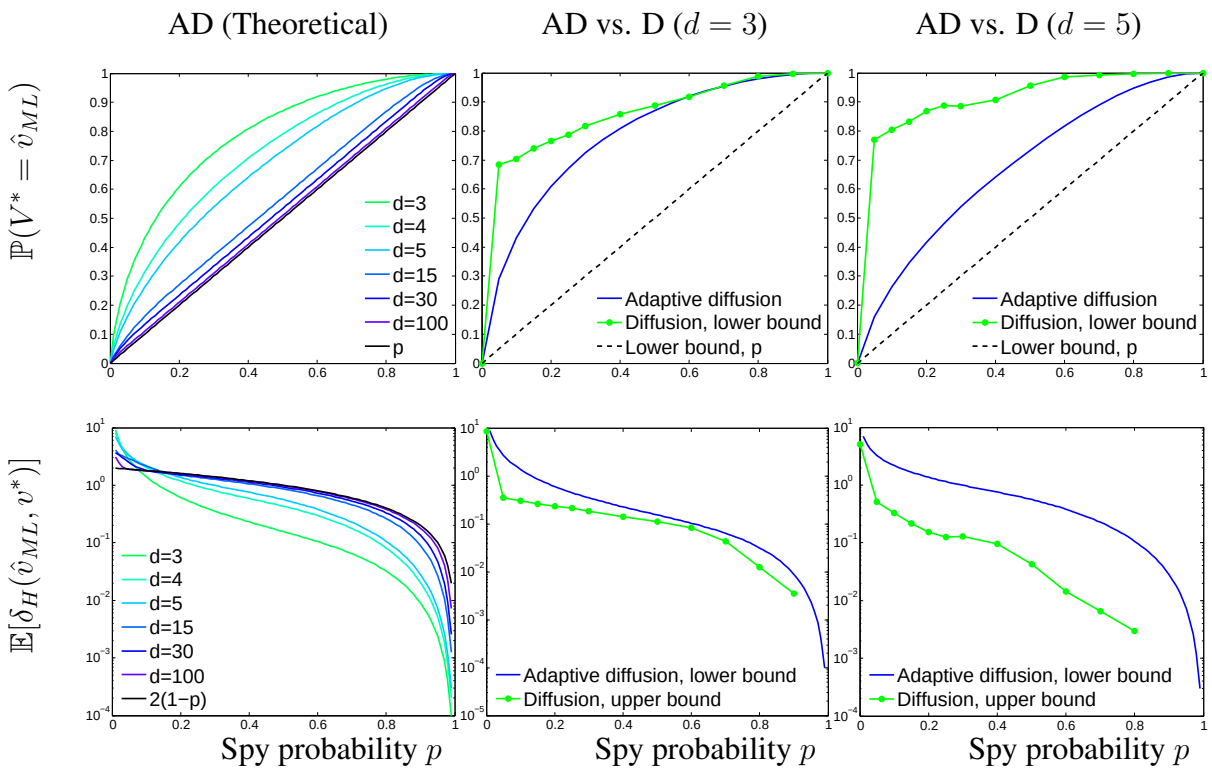


Figure 2.12: Adaptive diffusion (AD) theoretical performance for varying d (left). Adaptive diffusion improves over standard diffusion (D) and the gap increases as the degree of the underlying contact network increases (center, right).

The estimator then uses spies within the feasible subtree to prune out candidates. The goal is to identify nodes in the feasible subtree that are on the spine and close to the source. For each spy in the feasible subtree, there exists a unique path to the spine spy s_0, and at least one node on that path is on the spine; the spies' metadata reveals the identity and level of the spine node on that path with the lowest level—we call this node a pivot (details in Algorithm 4). For instance, in Figure 2.11, we can use spies 7 and 8 to learn that node 2 is a pivot with level $m_2 = 2$. Estimation hinges on the minimum-level pivot across all spy nodes, ℓ_{min}. In the example, $\ell_{min} = 1$, since spies 3 and 8 identify node 1 as a pivot with level $m_1 = 1$. The true source must lie in a subtree rooted at a neighbor of ℓ_{min}, with no spies. In our example, this leaves only node 0, the true source.

Anonymity properties. Using the described ML estimation procedure, we can exactly compute the probability of detection when adaptive diffusion is run over a d-regular tree.

Theorem 2.3.1. *Suppose the contact network is a regular tree with degree $d > 2$. There is a source node v^*, and each node other than the source is chosen to be a spy node i.i.d. with probability p as described in the spy model. Against colluding spies attempting to detect the location of the source,*

adaptive diffusion achieves the following:

(a) The probability of detection is

$$\mathbb{P}(\hat{v}_{ML} = v^*) = p + \frac{1}{d-2} - \sum_{k=1}^{\infty} \frac{q_k}{(d-1)^k} \ , \tag{2.26}$$

where $q_k \equiv (1-(1-p)^{((d-1)^k-1)/(d-2)})^{d-1} + (1-p)^{((d-1)^{k+1}-1)/(d-2)}$.

(b) The expected distance between the source and the estimate is bounded by

$$\mathbb{E}[\delta_H(\hat{v}_{ML}, v^*)] \geq 2 \sum_{k=1}^{\infty} k \cdot r_k \tag{2.27}$$

where $|T_{d,k}| = \frac{(d-1)^k-1}{d-2}$, *and*
$r_k \equiv \frac{1}{d-1}\Big((1-(1-p)^{|T_{d,k}|})^{d-1} + (d-1)(1-p)^{|T_{d,k}|} - (d-2)(1-p)^{|T_{d,k}|(d-1)} - 1\Big)$.

The proof is included in Section A. Briefly, it computes the probability of detection by conditioning on the lowest-level pivot node, ℓ_{min}. Given a pivot node, the probability of detection depends on the number of subtrees rooted at the neighbors of ℓ_{min} containing no spies.

Figure 2.12 illustrates the theoretical probability of detection and lower bound on expected distance from the true source as a function of the spy probability. We make two key observations:

Asymptotically optimal probability of detection: As tree degree d increases, the probability of detection converges to the degree-independent fundamental limit in Proposition 2.3.1, i.e., $\mathbb{P}(V^* = \hat{v}_{ML}) = p$. This is in contrast to diffusion, whose probability of detection tends to 1 asymptotically in d. The median Facebook user has 200 friends [101], so these asymptotics have practical implications, as we will see in Section 2.3.

Expected hop distance asymptotically increasing: We observe empirically that for regular diffusion, $\mathbb{E}[\delta_H(\hat{v}_{ML}, v^*)]$ approaches 0 as d increases. On the other hand, for adaptive diffusion with a fixed $p > 0$, as $d \to \infty$, $\limsup \mathbb{E}[\delta_H(\hat{v}_{ML}, v^*)] = 2(1-p)$. This holds because with probability $(1-p)$, the first node is not a spy, but with probability approaching 1 for d large enough, the first node on the spine will be a pivot node. Since the source is always a leaf, the distance from the estimate to the source will be at most 2 with probability approaching $(1-p)$. Figure 2.12 includes the line $2(1-p)$ for reference, and we observe that as $d \to \infty$, $\mathbb{E}[\delta_H(\hat{v}_{ML}, v^*)]$ appears to converge precisely to this line. However, for a fixed d, Theorem 2.3.1 implies that as $p \to 0$, $\mathbb{E}[\delta_H(\hat{v}_{ML}, v^*)] \to \infty$.

Comparison with diffusion. Here, we compare adaptive diffusion against traditional diffusion over d-regular trees. We have theoretical guarantees on the probability of detection under adaptive diffusion, but deriving such guarantees for regular diffusion is an open problem. We use a discrete-time diffusion process, in which each infected node passes the message to each neighbor with

Spy Probability p	Adaptive Diffusion	Diffusion ($q = 0.1$)	Diffusion ($q = 0.5$)
0.05	$\mathbf{0.10 \pm 0.03}$	0.83 ± 0.05	0.99 ± 0.01
0.10	$\mathbf{0.14 \pm 0.05}$	0.86 ± 0.05	1.00 ± 0.00
0.15	$\mathbf{0.17 \pm 0.05}$	0.90 ± 0.04	1.00 ± 0.00

Table 2.1: Probability of detection over the Facebook dataset [107], with 95% confidence intervals.

probability q in each timestep. As q increases, the variance of the associated geometric delay decreases, revealing the true source with higher probability. Since adaptive diffusion delays are deterministic, it is unclear how to fairly choose the variance of this delay. We thus considered a range of q values.

The ML source estimator for this spreading process is unknown. To lower bound the ML probability of detection, we consider two heuristic estimators: (1) the Gaussian estimator from [86], and (2) the first-spy estimator, which simply returns the parent of the first spy to observe the message. The estimator in [86] is ML when delays are i.i.d. Gaussian, whereas our delays are geometric. We nonetheless expect it to perform well for small p; since the distance between spies will be large, the delay distribution can be approximated by a Gaussian.

Figure 2.12 compares the probability of detection and expected hop distance for diffusion ($q = 0.7$) using heuristic estimators, against adaptive diffusion using the ML estimator. The lower bound on diffusion's probability of detection is the maximum of the simulated Pinto et al. estimator [86] and the first-spy estimator; the opposite holds for expected hop distance. For all p, adaptive diffusion performs better than diffusion, and the gap increases with degree. This effect is sensitive to q for small d, but we show in Section 2.3 that over real social graphs, the sensitivity to q becomes negligible.

Irregular Trees

Here, we consider tree networks with nonuniform node degrees. Regardless of spreading protocol or underlying network structure, the message always propagates over a tree superimposed on the underlying contact network. The probability of detection over irregular trees is therefore closely tied to performance over general graphs. ML estimation over irregular trees is more straightforward than in Section 2.2, primarily because we use the specialized Tree Protocol, which always places the source at a leaf node.

Proposition 2.3.3. *Suppose the underlying contact network $G(V, E)$ is an irregular tree with the degree of each node larger than one. One node $v^* \in V$ starts spreading a message at time $T = 0$ according to Protocol 3. Each node $v \in V$, $v \neq v^*$ is a spy with probability p. Let U denote the set of feasible candidate sources obtained by estimation Algorithm 4. Then the maximum likelihood*

estimate of v^* given U is

$$\hat{v}_{ML} = \arg\max_{u \in U} \frac{1}{\deg(u)} \prod_{v \in \mathcal{P}(u,\ell_{min}) \setminus \{u,\ell_{min}\}} \frac{1}{\deg(v) - 1}, \tag{2.28}$$

where ℓ_{min} is the lowest-level pivot node, $\mathcal{P}(u, \ell_{min})$ is the unique shortest path between u and ℓ_{min}, and $\deg(u)$ denotes the degree of node u. (Proof in Section A)

Real-World Networks

Equipped with a maximum-likelihood estimator for irregular trees, we can evaluate adaptive diffusion over a real dataset—namely, the Facebook social graph from [107] used in Section 2.2—against a spy-based adversary. We simulate adaptive diffusion and regular diffusion for $q \in \{0.1, 0.5\}$. We evaluate a lower bound on the probability of detection under diffusion using the first-spy estimator; for adaptive diffusion, we use a slightly-modified version of the ML estimator in Proposition 2.3.3, that accounts for cycles in the underlying graph. Table 2.1 lists the probability of detection averaged over 200 trials, for p up to 0.15.[1] Not only does adaptive diffusion hide the source better than diffusion, its probability of detection in practice is close to the fundamental lower bound of p. This is likely because the mean node degree in the dataset is 25, so high-degree asymptotics are significant. It is common for social networks to have degree distributions that skew large [25]. Additionally, while adaptive diffusion can never reach all nodes in a tree, cycles in the Facebook graph allow it to reach 81% of nodes within 20 timesteps.

2.4 Spy+snapshot adversarial model

Adversarial Model

The spy-based and snapshot adversarial models capture very different behavior. The spy-snapshot model considers a natural combination of both. At a certain time T, the adversary collects a snapshot of the infection pattern, G_T. It also collects metadata from all spies that have seen the message up to (and including) time T. Based on these two sets of metadata, the adversary infers the source.

Main Result (Spy+snapshot adversary)

Notably, this stronger model does not significantly impact the probability of detection as time increases. The snapshot helps detection when there are few spies by revealing which nodes are true leaves. This effect is most pronounced for small T and/or small p. The exact analysis of the probability of detection at T is given in Equation (2.29) in the Supplementary material, and Figure 2.13 illustrates how the effect of snapshot and spy nodes trade off.

[1] We chose this value because at its height, the Stasi employed 11 percent of the population as spies [63].

We follow closely the proof of Theorem 2.3.1 in Appendix A. Given a snapshot at a certain even time T, if there are at least two spy nodes infected at time T, then the adversary can perform the exact same estimation as he did with only spy nodes with $T \to \infty$. We only need to carefully analyze what happens when there is only one infected spy or no infected spies.

We condition on the lowest-level pivot node, ℓ_{min}, giving $\mathbb{P}(\hat{v}_{ML} = v^*) = \sum_{\ell_{min}} \mathbb{P}(\hat{v}_{ML} = v^* | \ell_{min}) \mathbb{P}(\ell_{min})$. Since ℓ_{min} lies on the spine, this is equivalent to conditioning on the distance of ℓ_{min} from the true source. We first define $|S_{d,T}| = 1 + d((d-1)^{T/2} - 1)/(d-2)$ as the number of nodes infected at time T, and $|\partial S_{d,T}| = d(d-1)^{(T/2)-1}$ as the number of leaves in the infected subtree. Then,

$$\begin{aligned}
&\mathbb{P}(\hat{v}_{ML} = v^*) = \\
&\underbrace{\frac{(1-p)^{|S_{d,T}|-1}}{|\partial S_{d,T}|}}_{\text{no spy}} + \sum_{k=1}^{T/2} \Bigg\{ \underbrace{\frac{(1-p)^{(|T_{d,k}|-1)}\, p}{|\partial T_{d,k}|}}_{\ell_{min}\ (k^{th} \text{ spine node}) \text{ is a spy}} + \\
&\underbrace{(1-p)^{|T_{d,k}|}(1-(1-p)^{|S_{d,T}|-|T_{d,k+1}|})\mathbb{E}_X\Big[\frac{\mathbb{I}(X \neq d-2)}{(X+1)\,|\partial T_{d,k}|}\Big]}_{\ell_{min}\ (k^{th} \text{ spine node}) \text{ not a spy}} + \\
&\underbrace{(1-p)^{|S_{d,T}|-(|T_{d,k+1}|-|T_{d,k}|)}\mathbb{E}_X\Big[\frac{\mathbb{I}(X \neq d-2)}{|\partial S_{d,T}| - (d-2-X)|\partial T_{d,k}|}\Big]}_{\text{all spy descendants of } k\text{-th spine node}} \Bigg\}, \qquad (2.29)
\end{aligned}$$

where $X \sim Binom(d-2, (1-p)^{|T_{d,k}|})$, $|T_{d,k}| = \frac{(d-1)^k - 1}{d-2}$ is the number of nodes in each candidate subtree for a pivot at level k, and $|\partial T_{d,k}| = (d-1)^{k-1}$ is the number of leaf nodes in each candidate subtree.

2.5 Connections to Pólya's urn processes

In this section, we make a connection between the adaptive diffusion on a line and the Pólya's urn process. In this process, we discover a property of Pólya's urn process, which provides a certain privacy. Further, we apply the Bayesian interpretation of the Pólya's urn process to design a new implementation of the adaptive diffusion and analyze the precise cost of revealing the control packets to the spy nodes.

To characterize the price of time stamps and the control packet separately, we focus on a concrete example of a line graph. Consider a line graph in which nodes 0 and $n+1$ are spies. One of the n nodes between the spies is chosen uniformly at random as a source, denoted by $v^* \in \{1, \ldots, n\}$. We let t_0 denote the time the source starts propagating the message according to some global reference clock. Let $T_{s_1} = T_1 + t_0$ and $T_{s_2} = T_2 + t_0$ denote the timestamps when the two spy nodes receive the message, respectively. Knowing the spreading protocol and the metadata, the adversary uses the maximum likelihood estimator to optimally estimate the source.

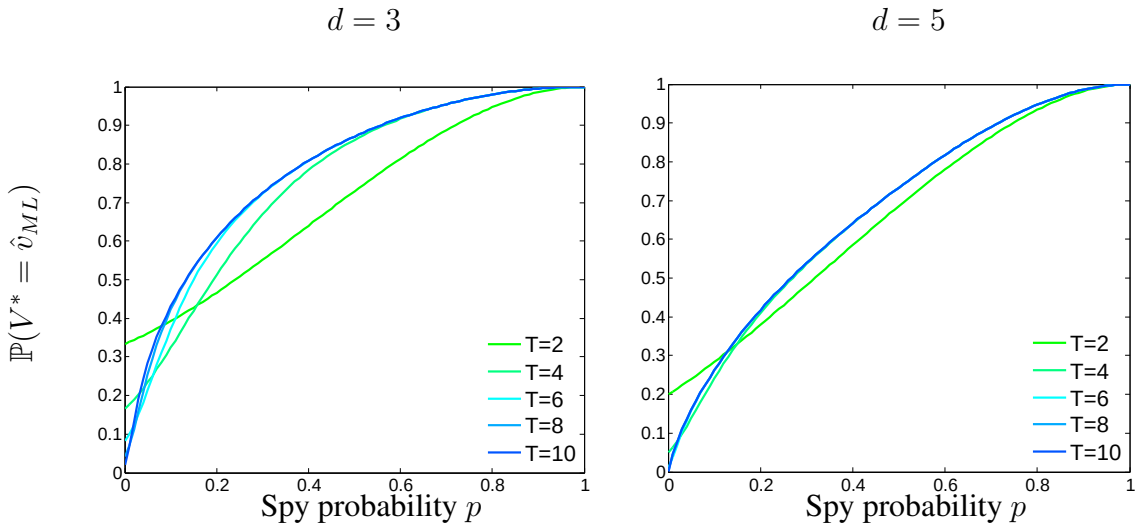


Figure 2.13: Probability of detection under the spies+snapshot adversarial model. As estimation time and tree degree increase, the effect of the snapshot on detection probability vanishes.

Standard diffusion. Consider a standard discrete-time random diffusion with a parameter $q \in (0, 1)$ where each uninfected neighbor is infected with probability q. The adversary observes T_{s_1} and T_{s_2}. Knowing the value of q, it computes the ML estimate $\hat{v}_{ML} = \arg\max_{v \in [n]} \mathbb{P}_{T_1 - T_2 | V^*}(T_{s_1} - T_{s_2} | v)$, which is optimal assuming uniform prior on v^*. Since t_0 is not known, the adversary can only use the difference $T_{s_1} - T_{s_2} = T_1 - T_2$ to estimate the source. We can exactly compute the corresponding probability of detection; Figure 2.14 (bottom panel) illustrates that the posterior (and the likelihood) is concentrated around the ML estimate, and the source can only hide among $O(\sqrt{n})$ nodes. The detection probability correspondingly scales as $1/\sqrt{n}$ (top panel).

Adaptive diffusion on a line. First, recall adaptive diffusion (Protocol 1) with the choice of $\alpha_d(h, t) = \frac{t-2h+2}{t+2}$ (Equation (2.5)) on a line illustrated in Figure 2.15. At $t = 0$, the message starts at node 0. The source passes the virtual source to node 1, so $v_2 = 1$. The next two timesteps ($t = 1, 2$) are used to restore symmetry about v_2. At $t = 2$, the virtual source stays with probability $\alpha_2(2, 1) = 1/2$. Since the virtual source remained fixed at $t = 2$, at $t = 4$ the virtual source stays with probability $\alpha_2(4, 1) = 2/3$. The key property is that if the virtual source chooses to remain fixed at the beginning of this random process, it is more likely to remain fixed in the future, and vice versa. This is closely related to the well-known concept of Pòlya's urn processes; we make this connection more precise later in this section.

The protocol keeps the current virtual source with probability $\frac{2\delta_H(v_t, v^*)}{t+2}$, where $\delta_H(v_t, v^*)$ denotes the hop distance between the source and the virtual source, and passes it otherwise. The control packet therefore contains two pieces of information: $\delta_H(v_t, v^*)$ and t.

Suppose spy nodes only observed timestamps and parent nodes but *n*ot control packets. The

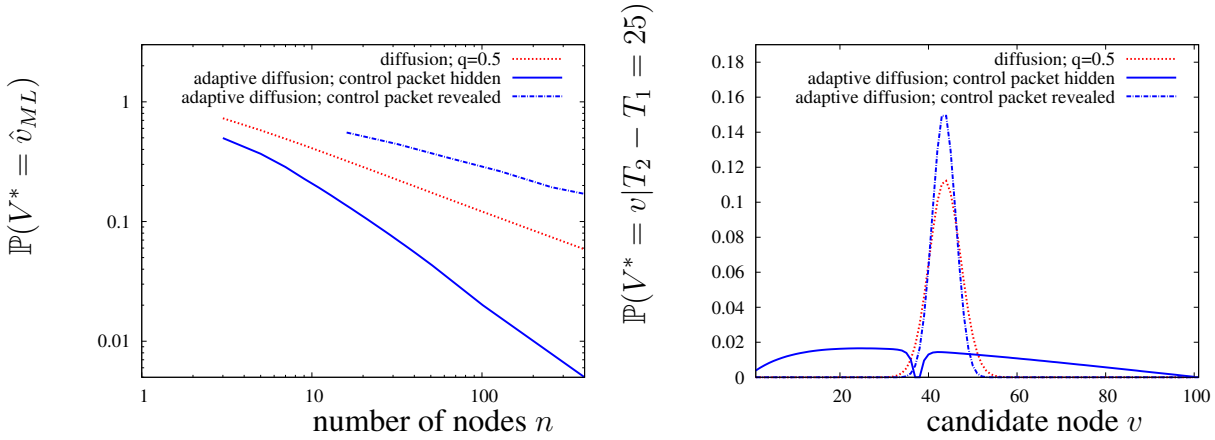


Figure 2.14: Comparisons of probability of detection as a function of n (top) and the posterior distribution of the source for an example with $n = 101$ and $T_2 - T_1 = 25$ (bottom). The line with 'control packet revealed' uses the Pólya's urn implementation.

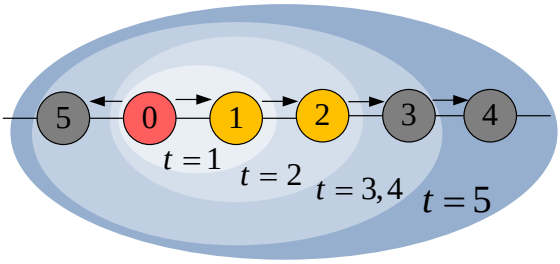


Figure 2.15: Spreading on a line. The red node is the message source. Yellow nodes denote nodes that have been, are, or will be the center of the infected subtree.

adversary could then numerically compute the ML estimate $\hat{v}_{ML} = \arg\max_{v \in [n]} \mathbb{P}_{T_1 - T_2 | V^*}(T_{s_1} - T_{s_2} | v)$. We can compute the corresponding detection probability exactly. Figure 2.14 shows the posterior is close to uniform (top panel) and the probability detection would scale as $1/n$ (bottom panel), which is the best one can hope for. Of course, spies do observe control packets, so they can learn $\delta_H(v^*, v_T)$ and identify the source with probability 1. We therefore introduce a new adaptive diffusion implementation that is robust to control packet information.

Adaptive diffusion via Pólya's urn. The random process governing the virtual source's propagation under adaptive diffusion is identical to a Pólya's urn process [61]. We propose the following alternative implementation of adaptive diffusion. At $t = 0$ the protocol decides whether to pass the virtual source left $(D = \ell)$ or right $(D = r)$ with probability half. Let D denote this random choice. Then, a latent variable q is drawn from the uniform distribution over $[0, 1]$. Thereafter, at each even time t, the virtual source is passed with probability q or kept with probability $1 - q$. It follows from the Bayesian interpretation of Pólya's urn processes that this process has the same

distribution as the adaptive diffusion process.

Further, in practice, the source could simulate the whole process in advance. The control packet would simply reveal to each node how long it should wait before further propagating the message. Under this implementation, spy nodes only observe timestamps T_{s_1} and T_{s_2}, parent nodes, and control packets containing the infection delay for the spy and all its descendants in the infection. Given this, the adversary can exactly determine the timing of infection with respect to the start of the infection T_1 and T_2, and also the latent variables D and q. A proof of this statement and the following proposition is provided in Section A. The next proposition provides an upper bound on the detection probability for such an adversary.

Proposition 2.5.1. *When the source is uniformly chosen from n nodes between two spy nodes, the message is spread according to adaptive diffusion, and the adversary has a full access to the time stamps, parent nodes, and the control packets that is received by the spy nodes, observations T_1, T_2, q and D, the adversary can compute the ML estimate:*

$$\hat{v}_{ML} = \begin{cases} \frac{T_1+2}{2} + \left\lfloor q\left(\frac{T_1-2}{2}\right)\right\rfloor & \text{, if } T_1 \text{ even and } D = \ell\,, \\ \frac{T_1+3}{2} + \left\lfloor q\left(\frac{T_1-1}{2}\right)\right\rfloor & \text{, if } T_1 \text{ odd and } D = \ell\,, \\ 1 + \left\lfloor (1-q)\left(\frac{T_1-1}{2}\right)\right\rfloor & \text{, if } T_1 \text{ odd and } D = r\,. \end{cases} \tag{2.30}$$

where T_1 is the time since the start of the spread until s_1 receives the message, and q is the hidden parameter of the Pólya's urn process, and D is the initial choice of direction for the virtual source. This estimator achieves a detection probability upper bounded by

$$\mathbb{P}\big(V^* = \hat{v}_{ML}\big) \leq \frac{\pi\sqrt{8}}{\sqrt{n}} + \frac{2}{n}. \tag{2.31}$$

Equipped with an estimator, we can also simulate adaptive diffusion on a line. Figure 2.14 (top) illustrates that even with access to control packets, the adversary achieves probability of detection scaling as $1/\sqrt{n}$ – similar to standard diffusion. For a given value of T_1, the posterior and the likelihood are concentrated around the ML estimate, and the source can only hide among $O(\sqrt{n})$ nodes, as shown in the bottom panel for $T_1 = 58$. In the realistic adversarial setting where control packets are revealed at spy nodes, adaptive diffusion can only hide as well as standard diffusion over a line.

2.6 Take-home Messages

In this chapter, we have presented adaptive diffusion. Adaptive diffusion is a message-spreading protocol that allows a user to anonymously spread a message over a network with provable anonymity guarantees against a variety of powerful, global adversaries. We considered two primary adversaries: a snapshot adversary and a spy-based adversary.

Under the snapshot adversary, we have shown that adaptive diffusion gives *perfect obfuscation* when the underlying network is a regular tree, and we precisely characterize the anonymity

properties when the network belongs to a class of irregular, random trees. We also observed empirically that nearly-optimal hiding holds even over realistic social graphs that are finite, irregular, and cyclic.

Under the spy-based adversary, we have shown that for regular trees, adaptive diffusion gives asymptotically optimal hiding in the degree of the underlying tree, whereas regular diffusion enables the adversary to locate the source with probability approaching one asymptotically in the degree of the tree. Again, we observed that these guarantees hold even for real-world graphs.

The fact that adaptive diffusion performs nearly as well on realistic graphs as on more stylized networks (namely, trees) is expected, because *any* message spreading pattern—regardless of spreading protocol—can equivalently be thought of as a tree superimposed on the underlying network. Therefore, our algorithm is designed to give optimal hiding on a worst-case graph. We can approximate a tree-based spread over a normal, cyclic network by choosing the adaptive diffusion spreading parameters carefully. For instance, by spreading a message to at most three neighbors at a time, the spreading pattern of adaptive diffusion over a typical cyclic, irregular social graph is reasonably approximated by a tree-structured network.

In terms of designing anonymous messaging platforms, this work is a concrete step towards defending against increasingly powerful adversaries that observe and participate in social networks to learn about their users. However, our work leaves several questions unanswered. We discuss some of these questions in Chapter 5, but the important point to keep in mind is that adaptive diffusion is only a small piece in a larger secure, privacy-preserving messaging ecosystem. Issues like namespace resolution and data transmission must be similarly secure and privacy-preserving to defend against realistic adversaries.

Chapter 3

Private Information Retrieval on Unsynchronized Databases

In the second part of this thesis, we propose algorithms for searching public data in a privacy-preserving fashion. As mentioned previously, we tackle this portion of the thesis in two steps: first, in this chapter, we consider the simpler problem of privately *retrieving* records from a database. In Chapter 4, we address the harder problem of executing privacy-preserving keyword queries on public databases.

Much research has been proposed to help people maintain privacy while searching databases. The most common class of solutions aims to hide the identity of the client, often through techniques like onion-routing; Tor is the most well-known solution in this class [35]. However, these systems work under the assumption that masking a client's identity gives a sufficient level of privacy. This may not be true in general, since the content of certain queries is strongly associated with the identity of the person making the query. For example, in 2009, AOL released anonymized search history data for its search engine. However, the correlation between queries and side information quickly led to deanonymization of users [7].

To address this concern, some private search tools instead mask the *contents* of web queries. These tools include chaffing and winnowing approaches like TrackMeNot [58], encrypted database searches [87], private information retrieval (PIR) [23], oblivious transfer (OT) [90], oblivious RAM [52], and private stream searches (PSS) [83, 17]. Some of these approaches are tailored to data retrieval, whereas others are tailored to keyword searches. In this chapter, we discuss the techniques related to data retrieval; in particular, we are interested in privacy-preserving retrieval that is robust to a common issue that arises in distributed systems: stale content on nodes.

This chapter is based on joint work with Kannan Ramchandran [43].

Related Work

The research community has considered a few privacy-preserving variants of data retrieval. One important variant is oblivious RAM (ORAM). In the ORAM problem, the client stores her data on a remote server and wishes to retrieve elements of this data without revealing to the server which data is being retrieved, while also protecting clients' data access patterns over time. It achieves this by storing data in encrypted form, and then shuffling the storage pattern of the data after each read operation [52, 85, 97, 104]. ORAM is not applicable to our setting because it assumes that the client owns the data in question, allowing her to encrypt the data with her own private key. In our setting, *many* clients are trying to access public data records that are hosted by a third party. As such, it would be impractical to assume the clients possess a shared key that is not visible to the adversarial server. More broadly, it deals with privately accessing encrypted data, whereas we are interested in privately accessing plaintext data.

Another variant of private data retrieval, which is the focus of our work, is called private information retrieval (PIR). In PIR, a client wishes to retrieve an indexed record from a plaintext database without revealing that index to the database server(s). This problem is closely related to k-out-of-n oblivious transfer [79]. The primary difference is that in PIR, it does not matter how much information the client learns about the database (as long as she retrieves her queried record), whereas in oblivious transfer, not only should the server not learn the client's query, but the client should also not learn anything about the database contents aside from the queried record. In this work, we assume the database is public, so the PIR setting is more appropriate than oblivious transfer.

Broadly, there are two types of PIR: single-server and multi-server. Single server PIR assumes a client-server architecture and provides computational security guarantees.[1] Single-server PIR tends to be computationally heavy, and often relies on expensive operations like modular exponentiation [100]. Multi-server PIR, on the other hand, tends to be more efficient (both in computation and communication) [81]. Despite its efficiency gains, multi-server PIR has gained little traction in practice, likely due to the restrictive assumptions on which it relies. The earliest forms of PIR relied on the following (implicit and explicit) assumptions [23]:

1. Multiple servers are available.
2. Each server stores a duplicate copy of the database.
3. The servers do not collude.
4. Servers are honest-but-curious.[2]
5. Servers willingly implement PIR algorithms.

[1] Trivial database transfer is the exception; it requires only one server and guarantees perfect privacy.

[2] An honest-but-curious adversary follows protocol, but tries to learn as much information as possible about private data held by other parties.

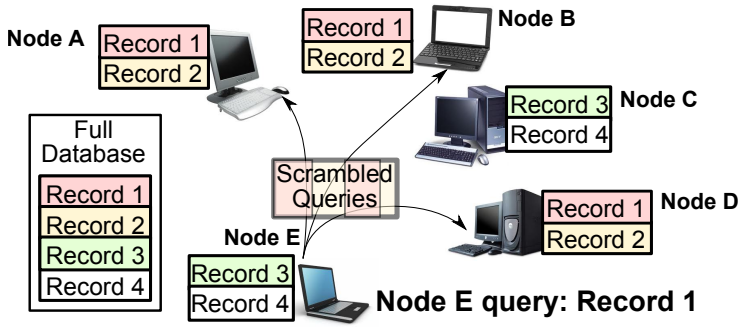


Figure 3.1: A P2P PIR system would organically circumvent several of the problems that arise in multi-server PIR.

Some multi-server PIR research has focused on relaxing assumptions 3 and 4. Existing schemes allow up to κ servers to collude without losing any privacy [13, 51]; other PIR schemes are robust to Byzantine servers that return arbitrary, incorrect information [12, 34]. Devet *et al.* observed that collusion-resistant and optimally Byzantine-resistant multi-server PIR can be computationally efficient in practice [34], and subsequently proposed a hybrid PIR scheme that combines ideas from computational and information-theoretic PIR [33].

Even with these promising developments, there are obstacles to widespread adoption. One of these is purely efficiency-related. The communication and computational costs of PIR are significant. A great deal of work has considered methods for reducing and characterizing fundamental bounds on the communication complexity of information-theoretic PIR [6, 14, 38, 110, 115, 116]; the state-of-the-art is the recent work by Dvir and Gopi, with a subpolynomial two-server scheme that relies on matching vector codes [37] . A smaller body of work considers methods for reducing the computational cost of multi-server PIR through techniques like precomputation [11] or batch codes [59, 70].

Another obstacle to adoption is that the assumptions listed above have not been fully addressed by existing literature. Namely, it seems unlikely that multiple servers would maintain identical databases without colluding. Further, there is little incentive for service providers like Google to allow private searches, barring legal intervention. We believe widespread collusion would be less likely in distributed systems, e.g. a peer-to-peer (P2P) setting. Privacy-minded individuals may sacrifice computational resources to help protect one another's privacy. We therefore envision storing small database chunks on different nodes in a P2P network and using these nodes as PIR servers (Figure 3.1). This architecture organically addresses assumptions 1, 3, 4, and 5, which suggests that multi-server PIR might be a useful tool in this setting. It also fits in nicely with our collaborative vision of user privacy.

However, to the best of our knowledge, neither a P2P architecture nor existing PIR research addresses assumption 2—indeed, a P2P architecture is likely to *increase* the likelihood of mis-synchronization between nodes, making assumption 2 even less plausible. Recent work has considered PIR schemes where servers store coded database chunks instead of identical copies [95, 44];

nonetheless, the coded database chunks are still computed from a unified view of the database. Our work departs from the existing literature by addressing assumption 2: we design a PIR algorithm that does not require the servers' databases to be identical.

Problem and Contributions

A client wishes to submit a multi-server PIR query to servers whose databases are not perfectly synchronized. That is, at least one server stores a different version of one or more records compared to the other servers (Figure 3.2). Our goal is to design a multi-server PIR scheme that is provably correct and private in this setting.

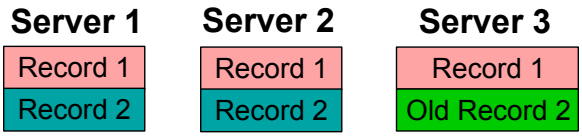


Figure 3.2: Existing multi-server PIR schemes fail when databases are unsynchronized—i.e., database order and size is preserved, but some records may be nonidentical across all servers.

In this chapter, we pose the problem of PIR over unsynchronized databases and recognize its connection to distributed source coding. We propose a two-stage architecture to solve the problem; this architecture leads to a collusion-resistant PIR scheme that probabilistically returns the desired record even when the contacted servers' databases are not perfectly synchronized. The key idea of our scheme is simple: we first determine which records are mis-synchronized, and then construct a PIR query that avoids these problematic records. When the number of unsynchronized database records scales sublinearly in the database size[3], our scheme has asymptotic communication and online computation costs that are identical to state-of-the-art PIR schemes. In practice, we incur slightly higher communication and server-side computation compared to traditional PIR, but we show through simulation that these costs are not prohibitive. Our approach also allows multiple queries to be processed in a single batch of PIR.

Outline

In the remainder of this chapter, we introduce some necessary background information on PIR and distributed source coding in Section 3.1. In Section 3.2, we present our proposed algorithm with theoretical privacy and efficiency guarantees. Section 3.3 provides experimental results obtained through simulation, and Section 3.4 reiterates the takeaway messages of this chapter.

[3]This assumption is reasonable for the applications we envision. In a P2P network, network nodes can sync databases with a 3rd party content provider frequently enough to ensure only a small number of records are unsynchronized on average between servers.

3.1 Background

We briefly explain private information retrieval and distributed source coding ideas related to our problem. Boldface lowercase variables denote vectors, and regular non-bolded variables denote scalars. Boldface uppercase variables denote matrices, and uppercase non-bolded, subscripted variables denote matrix elements. For $x \in \mathbb{Z}^+$, $[x]$ denotes the set $\{1, \ldots, x\}$.

Private Information Retrieval

Private information retrieval (PIR) is a technique allowing a client to retrieve the wth record from a database of n records ($w \in [n]$) without revealing the query index w to the server. As mentioned earlier, we will focus on multi-server PIR in this chapter, because it is significantly more efficient than single-server schemes in practice. Multi-server PIR gives information-theoretic privacy guarantees and assumes that the client has access to non-colluding servers with identical copies of the database.[4] We begin with an example of information-theoretic, multi-server PIR proposed by Chor et al. [23].

Basic PIR Scheme [23]

Two servers store identical copies of a database of records $\boldsymbol{f} = [f_1 \quad \ldots \quad f_n]^\intercal$, and a client wishes to retrieve the wth record, f_w. In practice, records can be of arbitrary length, but for simplicity, suppose each database element is a single bit, 0 or 1. The user's request can be represented by $\boldsymbol{e}_w \in \{0,1\}^N$, the indicator vector with a 1 at index w and 0's elsewhere. To disguise this query, the user generates a random string $\boldsymbol{a} \in \{0,1\}^N$ with each entry a Bernoulli$(1/2)$ random variable. The queries sent to servers 1 and 2 are $\boldsymbol{a} \oplus \boldsymbol{e}_w$ and $\boldsymbol{a}$, respectively. Each server computes the inner product of its received query vector with the database $\boldsymbol{f}$ using bitwise addition (XOR) and returns a single-bit result. The user XORs the results from the two servers to get f_w (see Figure 3.3). In an honest-but-curious adversarial model, this PIR scheme is information-theoretically private.

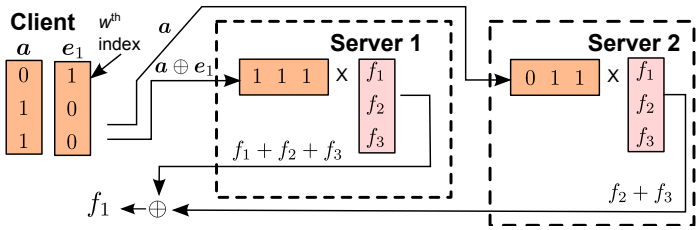


Figure 3.3: Two-server PIR scheme [23]. Each server computes the bitwise sum of a client-specified subset of database records. Since the two subsets differ only at the wth index, the binary sum of each server's results gives the desired record.

[4] Information-theoretic guarantees are secure against computationally unbounded adversaries.

Collusion Resistance

In multi-server PIR, privacy is lost if servers collude. There exist PIR schemes that offer κ-collusion-resistance—if no more than κ of the d servers collude, information-theoretic security is guaranteed [13, 51, 114]. We will explain a simplified version of [51], which is closely related to ideas from the area of multiparty computation [15, 96]. We later adapt this scheme to handle unsynchronized databases as well as concurrent queries.

The basic idea of collusion-resistant PIR in [51] is as follows: The client designs a random polynomial $r(X)$ of degree κ, such that $r(0) = f_w$ (the desired record). The client instructs each server to evaluate $r(X)$ at a distinct $X \neq 0$. The client then interpolates these evaluations to recover $r(0)$. Since $\kappa + 1$ points are required to interpolate a κ-degree polynomial, up to κ servers can collude without compromising privacy.

More detailed explanation: In the two-server case, the client transmitted $\boldsymbol{a}$ and $\boldsymbol{a} \oplus \boldsymbol{e}_w$ to servers 1 and 2, respectively. These queries can be interpreted as two characteristic points from a linear, vector-valued polynomial $\boldsymbol{q}(X) = \boldsymbol{a}X + \boldsymbol{e}_w$. Namely, when $X = 1$, we get $\boldsymbol{q}(1) = \boldsymbol{a} + \boldsymbol{e}_w$, and $\boldsymbol{q}'(1) = \boldsymbol{a}$. We use $r(X)$ (or $r'(X)$) to denote a server's reply to the PIR query $\boldsymbol{q}(X)$ (or $\boldsymbol{q}'(X)$, respectively). The client's goal is to determine $r(0)$, the output for the query $\boldsymbol{q}(0) = \boldsymbol{e}_w$. It is straightforward to show that $r(X)$ is a linear function of X and $r(0) = f_w$, so we need only two distinct input queries to interpolate the value of $r(0)$. In the two-server example, $r(0)$ is interpolated by taking $\hat{r}(0) = r(1) + r'(1)$. This two-server query is not immune to server collusion.

Now suppose we wish to resist collusion between any set of at most κ servers in the system—for concreteness, let $\kappa = 2$. We can achieve this by querying $d \geq \kappa + 1$ servers; for our concrete example, let $d = 3$. Instead of the linear query from the two-server case, we now submit query vectors that are quadratic in X, i.e. of degree κ. Note that this scheme requires a finite field $GF(2^\ell)$ containing at least $\kappa + 2$ elements. So our example polynomial query is $\boldsymbol{q}(X) = \boldsymbol{a}X^2 + \boldsymbol{b}X + \boldsymbol{e}_w$, where $\boldsymbol{a}, \boldsymbol{b} \in \{0, \ldots, 2^\ell\}^n$ are elementwise randomly-drawn vectors, and $\boldsymbol{e}_w$ is a vector of zeros with a one at index w. So if the database has two elements, this is equivalent to drawing $n = 2$ independent, univariate, κ-degree polynomials; all the n polynomials have a zero constant term except for the polynomial corresponding to the desired record index w. Concretely, suppose we want record $w = 1$. Then we draw two random quadratic polynomials (one per database record).

$$\boldsymbol{q}(X) = \begin{bmatrix} q_1(X) \\ q_2(X) \end{bmatrix} = \begin{bmatrix} a_1X^2 + b_1X + 1 \\ a_2X^2 + b_2X + 0 \end{bmatrix}, \tag{3.1}$$

where a_i, b_i, q_i denote the ith elements of vectors $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{q}$, respectively. The client evaluates these n polynomials (one for each database element) at d distinct values of X (one for each server), and compounds the results into query vectors. For instance, the client might send $\boldsymbol{q}(1)$ to server 1, $\boldsymbol{q}(2)$ to server 2, and $\boldsymbol{q}(3)$ to server 3. Since the queries are quadratic in X, at least three servers must collude to learn the desired record. Upon receiving such a query, each server projects the database

onto the query, and returns a single value as a result:

$$\begin{aligned} r(X) &= \boldsymbol{q}(X)^\intercal \boldsymbol{f} \\ &= f_1(a_1X^2 + b_1X + 1) + f_2(a_2X^2 + b_2X) \\ &= (f_1a_1 + f_2a_2)X^2 + (f_1b_1 + f_2b_2)X + f_1 \end{aligned}$$

Observe that $r(X)$ is quadratic in X, and $r(0) = f_w$. The client therefore needs three distinct function points to interpolate $r(0)$, or $\kappa + 1$ points in general.[5] We write

$$\underbrace{\begin{bmatrix} r(1) \\ r(2) \\ r(3) \end{bmatrix}}_{\boldsymbol{r}} = \underbrace{\begin{bmatrix} 1^2 & 1^1 & 1^0 \\ 2^2 & 2^1 & 2^0 \\ 3^2 & 3^1 & 3^0 \end{bmatrix}}_{\mathbf{V}} \times \begin{bmatrix} \boldsymbol{a}^\intercal \\ \boldsymbol{b}^\intercal \\ \boldsymbol{e}_w^\intercal \end{bmatrix} \times \begin{bmatrix} f_0 \\ \dots \\ f_n \end{bmatrix}$$

which is equivalent to $\boldsymbol{r} = \mathbf{V} \times [\ \cdots\ r(0)]^\intercal$. The interpolated $\hat{r}(0) = f_w$ can be computed using the bottom row of $\mathbf{V}^{-1}$ in the above expression ($\mathbf{V}$ is full-rank). In our example, we get $\hat{r}(0) = [0\ \ 0\ \ 1] \cdot \mathbf{V}^{-1}\boldsymbol{r} = r(3) + 3r(2) + 3r(1)$. This approach can be extended to accommodate arbitrary collusion-resistance parameters (Algorithm 9, Appendix C.2).

Distributed Source Coding

Our approach to PIR relies on distributed source coding, in which multiple, non-communicating sources attempt to efficiently communicate correlated information to a receiver. In our problem, the client is the receiver, and the servers are the distributed sources. We assume the number of unsynchronized database elements s is small, so the servers' contents are highly correlated. The client wishes to learn which database elements are unsynchronized—i.e., the difference of the servers' data—to successfully complete PIR. Concretely, suppose d servers—say $d = 2$—store databases $\boldsymbol{f}^{(1)}$ and $\boldsymbol{f}^{(2)}$ that are differentially sparse, i.e. the vector $\boldsymbol{r}(0) = |\boldsymbol{f}^{(1)} - \boldsymbol{f}^{(2)}|$ is comprised mostly of zeros.[6] The goal is for the client to learn which records are unsynchronized with minimal computation and communication; this is equivalent to learning the support of the sparse vector $\boldsymbol{r}(0)$. This problem has been studied for $\boldsymbol{f}^{(i)} \in \{0,1\}^n$ by Korner and Marton [65].

Initially, we ignore the distributed servers and instead suppose a genie has access to $\boldsymbol{r}(0)$. The genie must minimize the communication and computation to communicate $\boldsymbol{r}(0)$ to the client. This is equivalent to finite-field compressed sensing. We will show two approaches for solving the problem with a genie, after which we will explain how a client can learn the support of $\boldsymbol{r}(0)$, which encodes the mis-synchronized database indices, from distributed, non-colluding servers at nearly the same cost as as that incurred by the genie.

[5] In this example, we could actually decode with only two replies—$r(X)$ is linear in f_1 and f_2. This special case arises because the database size $n = 2$.

[6] Our notation here is similar to that previously used for server response polynomials, $r(X)$. This is because we will soon consider vector-valued server response polynomials $\boldsymbol{r}(X)$ that are sparse when evaluated at $X = 0$.

Genie-dependent solutions

$\boldsymbol{r}(0)$ contains n elements, and at most s of them are nonzero, $s \ll n$. The genie can send linear combinations of the entries of $\boldsymbol{r}(0)$, called parity symbols (or measurements in compressed sensing literature) to the client for decoding. These parity symbols are generated by left-multiplying the sparse vector by a parity check matrix $\mathbf{A}$ (or sensing matrix in compressed sensing), giving a measurement vector $\boldsymbol{y}$, as illustrated in Figure 3.4. If $\mathbf{A}$ is well-designed, $\boldsymbol{r}(0)$ can be reconstructed from few parity symbols; $\mathbf{A}$ plays a compressive role, reducing genie-to-client communication.

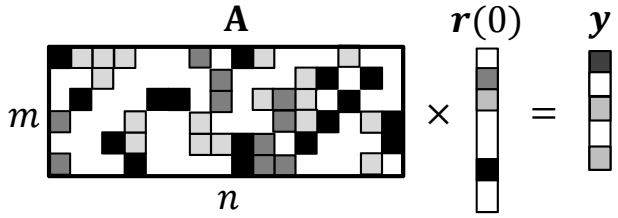


Figure 3.4: Compression setup. A sparse vector $\boldsymbol{r}(0)$ can be fully reconstructed with the $m < n$ samples in $\boldsymbol{y}$ if parity check matrix $\mathbf{A}$ is well-designed.

MDS codes: One approach for reconstructing the desired vector $\boldsymbol{r}(0)$ uses maximum distance separable (MDS) codes, such as Reed-Solomon (RS) codes. An MDS code with message size k and block length b can correct the theoretical maximum of $\lfloor (b-k)/2 \rfloor$ errors [91]. So if a received codeword is corrupted with s errors (the wrong symbol is received), then as long as $2s < (b-k-1)$, an MDS code can recover the initial message. Additionally, in a systematic MDS code, the first k symbols of the codeword are exactly the k message symbols. The remaining $b-k$ symbols are known are parity symbols, which are linear combinations of the message symbols.

To use MDS codes for our problem, we exploit the sparsity of $\boldsymbol{r}(0)$. Encoding $\boldsymbol{r}(0)$ with a systematic RS code that corrects at most s errors, the resulting codeword would be $[\ \boldsymbol{r}(0)^\intercal\ g_1\ g_2\ \ldots\ g_{2s}\ ]$, where the g_i's denote parity symbols. The receiver measures only the $2s$ parity symbols (it assumes the first n systematic symbols are all zeros due to our sparsity assumption). Then the recorded codeword would contain exactly s errors, all in the systematic symbols; the decoder could correct them all and deterministically recover the nonzero entries of $\boldsymbol{r}(0)$. This scheme has low communication cost, but the decoding complexity can be prohibitive in practice—particularly if the number of unsynchronized records is large. For instance, Berlekamp-Massey decoding for RS codes has asymptotic complexity $O((n+2s)^2)$ for a codeword of block length $n+2s$ [56]. Faster algorithms exist, but are still superlinear in n.

A probabilistic approach: A different scheme called PULSE is well-suited to reconstructing $\boldsymbol{r}(0)$ when the number of nonzero entries is large [84]. Intuitively, the encoder maps linear combinations of the sparse vector entries into a small number of bins; the decoder is able to recover the original entries by iteratively peeling nonzero entries from the bins, thereby efficiently solving a linear system of equations. The parity check matrix $\mathbf{A}$ in this scheme is constructed as $\mathbf{A} = \mathbf{F} \otimes_r \mathbf{H}$, where $\otimes_r$ denotes a row-tensor product, and $\mathbf{F}$ is the first three rows of an $n \times n$ Vandermonde ma-

trix. $\mathbf{H}$ is a binary low-density parity-check (LDPC) matrix, designed using a left-regular LDPC construction for simplicity (an optimal irregular construction can also be used [69]). A left-regular construction means that each column in the LDPC matrix has the same number of nonzero entries.

The LDPC matrix $\mathbf{H}$ can be thought of as a bipartite graph between the elements of the sparse vector $\boldsymbol{r}(0)$ and the measured bins (Figure 3.5). If a bin in $\boldsymbol{y}$ maps to exactly one nonzero element of $\boldsymbol{r}(0)$, then we call that bin a singleton; if more than one nonzero element maps to a bin in $\boldsymbol{y}$, that element is a multiton. Otherwise, the bin is a zeroton.

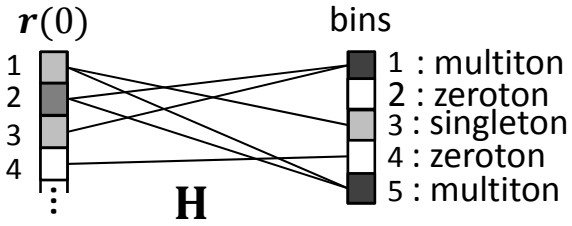


Figure 3.5: $\mathbf{H}$ maps elements of $\boldsymbol{r}(0)$ to bins. Singleton bins map to a single nonzero element, while multitons map to multiple nonzero elements. On finding a singleton, the decoder strips it from all bins. For example, by stripping element 1 of $\boldsymbol{r}(0)$ from bins 3 and 5, bin 5 becomes a singleton. After row-tensoring $\mathbf{H} \otimes_r \mathbf{F}$, each bin has two additional entries (not pictured) that help the decoder decide if a bin is a singleton.

Matrix $\mathbf{F}$ helps the decoder learn which bins are singletons. The singleton detection algorithm is detailed in Appendix C.2, Algorithm 10. Intuitively, after finding a singleton, the decoding algorithm subtracts the corresponding nonzero element of $\boldsymbol{r}(0)$ from all elements of y to which it maps. The components of this scheme are well-studied, but this particular scheme over finite fields has not been explicitly described in the literature. For completeness, we include the following result:

Theorem 1. *(From [84]) Suppose the compressed sensing algorithm above uses a degree-3, left-regular LDPC construction, with $m > 3s \cdot 1.23$ measurements. If the signal sparsity scales according to $s = n^{\delta}$, where $\delta \leq 1/3$, the peeling decoder asymptotically recovers the vector support with probability at least $1 - O(\frac{1}{m})$ in $O(s)$ operations.*

The proof can be found in [84]. The $\delta \leq 1/3$ sparsity constraint is not restrictive because our PIR scheme is only practical for small numbers of mis-synchronizations; nonetheless, PULSE can handle essentially any sublinear sparsity scaling if $m > 6s$. These results are asymptotic, but in practice, PULSE performs well even on small databases.

Distributed solution

So far, we assumed the genie has access to sparse vector $\boldsymbol{r}(0)$. However, servers' databases are only assumed to be differentially sparse, i.e. their difference is sparse, and those servers cannot

d	Number of servers contacted
$\alpha_1, \ldots, \alpha_d$	Values at which query polynomial is evaluated
κ	Maximum number of colluding servers
n	Size of the database (in records)
s	Maximum number of unsynchronized records
$\mathbf{A}$	Parity check matrix, $m \times n$
m	Number of parity symbols
L	Record size (bits)
$\ell,\ \ell_2$	Field sizes in Phase 1 ($GF(2^\ell)$) and Phase 2 ($GF(2^{\ell_2})$), respectively
$f_1, \ldots, f_\zeta$	Desired records
$u_1, \ldots, u_s$	Indices of unsynchronized records

Table 3.1: System parameters

communicate with each other. A result by Korner and Marton [65] states that to learn the differential of the two databases over a binary field with minimum communication, each server should encode its local view of the database using a random linear code with rate $H_b(|\boldsymbol{f}^{(1)} - \boldsymbol{f}^{(2)}|)$, where H_b denotes the binary entropy function. We cannot utilize the same solution because of decoding complexity and our need for collusion-resistance (requiring non-binary fields). However, our approach uses a similar intuition, much like [88]. Each server individually encodes its database with the described linear codes. We make no claims of optimality, but we will show later that the overhead is small. This approach is conceptually similar to Biff codes for set reconciliation [78]; however, we consider the different problem of sequence reconciliation.

Nonetheless, we can use the Korner-Marton result to lower-bound communication costs. For instance, if approximately 11 percent of the database is unsynchronized and each record is one bit, each server must send at least $\frac{n}{2}$ bits; between two servers, the client is receiving an entire database's worth of communication. This observation—and more generally, the concavity of the binary entropy function in the Korner-Marton result—supports our assertion that the number of unsynchronized records must be kept small.

3.2 Algorithm Description

Existing PIR methods are provably correct when the servers store identical copies of the database. However, individual server nodes may store unsynchronized versions of a database. This will cause traditional PIR algorithms to fail with high probability. For example, consider the two-server PIR scheme in section 3.1 when one server has an out-of-date entry. When the client decodes the replies from the servers, she gets the sum of the desired record with an error term (Figure 3.6).

There are a few ways to circumvent this issue. One is to treat the reply from the out-of-date server as an error, and query another server. Schemes like [34, 51] inherently enable this via robustness to Byzantine servers. However, it may be more expensive to query a new server than to communicate more with an already-connected server. It may also be unrealistic to assume that the client will find any set of perfectly-synchronized servers in a reasonable amount of time.

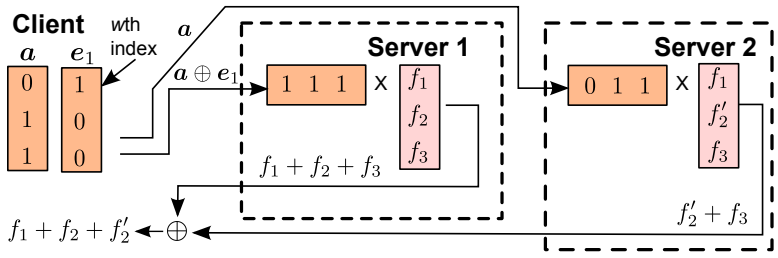


Figure 3.6: PIR when the databases are unsynchronized; server 2 has an out-of-date record, f_2'. Traditional PIR methods fail in this scenario. In this example, the desired record was f_0, but the client received the summation of f_1 and an error term, $f_2 + f_2'$.

We propose a scheme that functions over unsynchronized databases at the expense of increased computation and communication. When the number of unsynchronized records is small, we can treat PIR as a distributed source coding problem. We assume throughout that synchronization errors are i.i.d. and uniformly distributed across the database.

Main Idea: Our proposed scheme is divided into two communication rounds. In Phase 1, we learn which records are unsynchronized; in Phase 2, we conduct PIR knowing the synchronization error locations. We adapt [51], a collusion-resistant scheme for synchronized databases (section 3.1).

Phase 1: Locate unsynchronized records

In this phase, the objective is to efficiently learn the mis-synchronization locations from distributed servers. The client transmits no information about the desired record index, so privacy is preserved. Suppose the client knows that at most s records are unsynchronized. Unlike [42], we assume that the database stores hashes of each record—if two servers have the same version of record f_i, then they both store the same hash $H(f_i)$; otherwise, the hashes are different with high probability [20]. We will show that this lowers our accuracy slightly compared to [42] in exchange for communication costs that are logarithmic in record size instead of linear.

Basic Solution

Suppose we have only two servers, and a genie sums the servers' respective views of the database hashes over $GF(2^\ell)$, giving $H(\boldsymbol{f}^{(1)}) + H(\boldsymbol{f}^{(2)})$.[7] The synchronized (i.e., equal) hashes cancel, giving a sparse vector of length n, with nonzero entries at the unsynchronized records. A parity check matrix $\mathbf{A}$ could be used to compress this sparse vector for transmission to the client: $\mathbf{A} \cdot (H(\boldsymbol{f}^{(1)}) \oplus H(\boldsymbol{f}^{(2)}))$. By linearity, this is equivalent to $\mathbf{A} \cdot H(\boldsymbol{f}^{(1)}) \oplus \mathbf{A} \cdot H(\boldsymbol{f}^{(2)})$. So to communicate the same information in a distributed fashion, each server can simply compress its own database with a pre-determined $\mathbf{A}$ matrix, and the client can recover the sparse vector from the compressed vectors.

[7] We use $H(\boldsymbol{f})$ to denote the elementwise hash of vector $\boldsymbol{f}$.

The same idea works for many databases: each server S_i individually compresses its view of the database by returning $\mathbf{A} \cdot \boldsymbol{f}^{(i)}$ to the client. The client can then do pairwise reconstruction, finding d sets of unsynchronized records between $(S_1, S_2), (S_2, S_3), \ldots,$ and (S_{d-1}, S_d). The final answer is the union of these sets. This approach requires the client to compute d distinct sparse vector reconstructions.

Better Solution

It is possible to offload slightly more computation to the servers, so the client only needs to compute one sparse reconstruction. We use this approach in simulation. Suppose the client generates a scalar query polynomial $q(X)$ such that $q(0) = 0$ and the set of roots of q is disjoint from the set $\{\alpha_1, \ldots, \alpha_d\}$. The client uses this polynomial for all the database records, and sends $q(\alpha_1),\ q(\alpha_2), \ldots,\ q(\alpha_d)$, to the d servers. Upon receiving a query $q(X)$, each server returns to the client the $n \times 1$ vector $\boldsymbol{r}(X) = q(X) \cdot H(\boldsymbol{f})$—a scaled version of the hashed database. Note that servers are not (yet) compressing the replies.

The client interpolates the length-n reply vectors just as in section 3.1 to get $\hat{\boldsymbol{r}}(0)$, so $\hat{\boldsymbol{r}}(0)^\intercal = \underbrace{[0 \ \ldots \ 0 \ 1]}_{1\times d} \cdot \mathbf{V}^{-1}\mathbf{R}$, where $\mathbf{R}$ is a $d \times n$ matrix: $\mathbf{R} = \left[\ \boldsymbol{r}(\alpha_1) \ \ldots \ \boldsymbol{r}(\alpha_d)\ \right]^\intercal$. If the databases were perfectly synchronized, we would have $\hat{\boldsymbol{r}}(0) = \mathbf{0}_n$.

Lemma 1. *Suppose f_j is synchronized across databases (i.e., all of the servers have identical versions of the record). Then $\hat{\boldsymbol{r}}(0)_j = 0$. Otherwise, for any $c > 0$, $\hat{\boldsymbol{r}}(0)_j \neq 0$ with probability at least $1 - 1/L^c$, where L denotes record size (bits). Increasing c reduces the probability of error geometrically in L at the expense of a linear increase in downlink communication, i.e., $nc\log_\ell(L)$ symbols per server*
(Proof in Appendix B.1).

So if at most s records are unsychronized across all servers (assume s is known), there will be exactly s nonzero entries in $\hat{\boldsymbol{r}}(0)$ at the unsynchronized indices (e.g. index 2 in Figure 3.7) with probability $\geq (1 - 1/L^c)^s$. The client can thus learn which records are unsynchronized. If we choose $c = \log_L n$, the probability of success is lower bounded by $1 - s/n$. When s is sublinear in database size n, the probability of success asymptotically approaches 1. The error stems from using hashes of records. If we use entire records as in [42], decoding succeeds deterministically, but with communication costs that are linear in record size L instead of logarithmic.

Reducing communication: In the previous solution, each server sent $O(n)$ bits for the client to learn the support of $\hat{\boldsymbol{r}}(0)$. However, for $s \ll n$, $\hat{\boldsymbol{r}}(0)$ will be a sparse vector. We can therefore reduce the downlink communication by using distributed compression. In $\hat{\boldsymbol{r}}(0)$, nonzeros at the mis-synchronized indices are like errors in a codeword, which require two parity symbols to locate and correct (if the hash length L_h is more than one symbol, this can be adapted for bursty error correction with s bursts). Suppose each hash requires one symbol. The elements of $\hat{\boldsymbol{r}}(0)$ can be

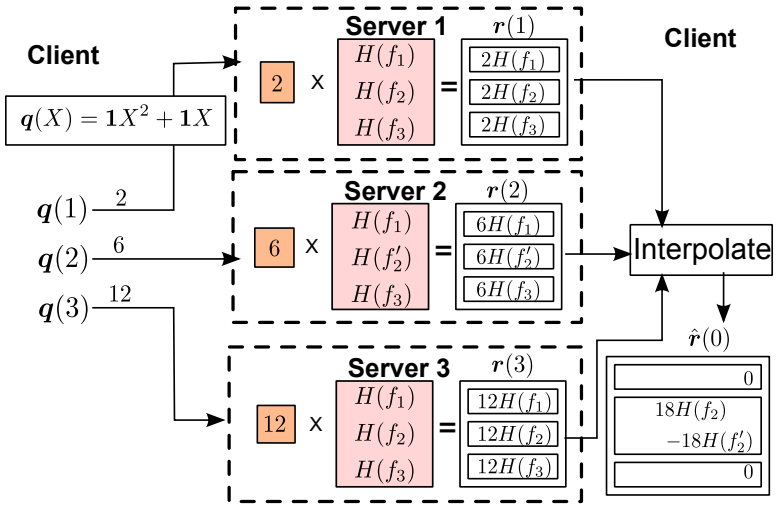


Figure 3.7: Identifying unsynchronized records without any compression. f_2 is not synchronized across all servers, so the 2nd entry of $\hat{\boldsymbol{r}}(0)$ is nonzero. This figure is computed over integers, but in practice, all operations are over finite field $GF(2^\ell)$.

recovered deterministically with $2s$ samples using MDS codes, or with high probability by measuring more than $3.69s$ samples according to PULSE compressive sensing using left-regular LDPC codes [84]. Although none of the servers knows the sparse vector $\hat{\boldsymbol{r}}(0)$, each server can individually compress its reply vector, and the client can recover the sparse vector. The client decodes the servers' replies to obtain vector $\boldsymbol{y}$, which is decoded according to Algorithm 4 (Appendix C.2). Algorithm 5 specifies the procedure for identifying unsynchronized records.

Lemma 2. *Suppose matrix* $\mathbf{A}$ *in Algorithm 5 is constructed from the parity check matrix of a systematic, MDS* $(n + 2s, n, 2s + 1)$ *error-correcting code. If there are at most* s *unsynchronized records, the locations of* s_o *unsynchronized records* ($s_o \leq s$) *can be recovered with probability greater than* $1 - s_o/L^c$*, assuming synchronization errors are i.i.d. and uniformly random (Proof in Appendix B.1).*

If there are many mis-synchronizations, Reed-Solomon decoding can be inefficient, so PULSE decoding (or another non-MDS code with fast decoding) may be more appropriate.

Protocol 5 A κ-private algorithm for learning the locations of up to s unsynchronized records.

Client:

1: Choose d distinct indices $\alpha_1, \ldots, \alpha_d$ from $GF(2^\ell)$. Define matrix $\mathbf{V}$ as $V_{ij} = \alpha_{d-i+1}^{\kappa-j+1}$ (to be used later).
2: Define the query polynomial as
3: $q(X) = X^\kappa + X^{\kappa-1} + \ldots + X$.
4: Send $q(\alpha_i)$ and s to server S_i, $i \in [d]$.

Each honest-but-curious server (S_i, $i \in [d]$)**:**

5: Select $\mathbf{A}$ as the first $2s$ rows of a global parity check matrix $\overline{\mathbf{A}}$, known to all servers.
6: Using $q(\alpha_i)$, compute $\tilde{\boldsymbol{r}}(\alpha_i) = q(\alpha_i)\mathbf{A}H(\boldsymbol{f}^{(i)})$.
7: Return $\tilde{\boldsymbol{r}}(\alpha_i)$ to the client.

Client:

8: On receiving $\tilde{\boldsymbol{r}}(\alpha_i)$, build $\tilde{\mathbf{R}} = [\tilde{\boldsymbol{r}}(\alpha_1) \ldots \tilde{\boldsymbol{r}}(\alpha_d)]$.
9: Compute $\boldsymbol{y} = \tilde{\mathbf{R}} \cdot (\mathbf{V}^{-1})^\intercal \cdot [0 \;\; 0 \;\; \ldots \;\; 1]^\intercal$.
10: Reconstruct $\hat{\boldsymbol{r}}(0)$ from $\boldsymbol{y}$ by decoding.
11: Return the support of $\hat{\boldsymbol{r}}(0)$ as the unsynchronized record indices.

Corollary 2. *Suppose desired record f_w is synchronized across all contacted servers. Suppose Algorithm 5 is run with sensing matrix $\mathbf{A}$ chosen according to §3.1 [84]. If the number of unsynchronized records is upper bounded by $n^{1/3}$, then the PIR query asymptotically succeeds with probability at least $(1 - O(1/m))(1 - \frac{s_o}{L^c})$, where m is the number of measured bins, $m > 3.69s$ (Proof in Appendix B.1).*

Efficiency: Each server can precompute a compressed version of the hashed database using a worst-case estimate of the number of unsynchronized records s. Then the client need only send the polynomial evaluation point $q(X)$ and s (estimated via network measurement) to each server. Upon receiving such a query, the server can extract the first $2s$ entries from its compressed database, scale them by $q(X)$, and return the result. Using this scheme, a client can learn the unsynchronized record locations in one round of communication, with at most $d(\log_2 d + 2sc \log_2 L)$ total bits of communication.

Phase 2: Retrieve the desired record(s)

In the second round of communication, the client retrieves the desired records using a modified version of existing PIR schemes. The client now knows which records are unsynchronized from Phase 1, so it must ensure that the servers avoid touching those records. For example, suppose the client learns that record i is unsynchronized. In two-server PIR, both servers' query vectors should be zero at index i so neither server touches the unsynchronized record. The same idea holds for more servers; the random query polynomial for the ith (unsynchronized) record should have roots at $\{\alpha_1, \ldots, \alpha_d\}$ so that none of the servers touches the unsynchronized record. Since the query polynomials have degree $< d$, this condition holds only for the zero polynomial. However, the

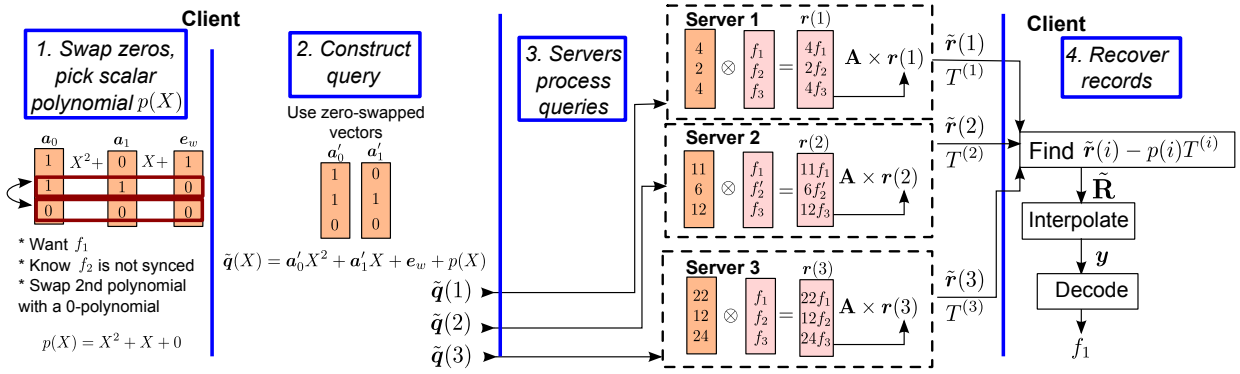


Figure 3.8: Example PIR algorithm, Phase 2, on a database with $n = 3$ records. The client knows f_2 is unsynchronized, so it swaps zeros into the second query index before adding $p(X)\mathbf{1}_n$ to each query vector. The client wants one record, so $\mathbf{A}$ is the first row of a Vandermonde matrix, $\mathbf{1}_n^\intercal$. Thus 'decoding' means multiplying the results by $\overline{\mathbf{V}}^{-1} = [1]^{-1}$ (Algorithm 6).

client cannot just set the ith query polynomial to zero, as this would leak information. Instead, for each nonzero query polynomial at a mis-synchronized index, we swap it with a zero polynomial found elsewhere in the query. Since query polynomials are drawn i.i.d. uniformly at random, the classical result on exchangeability of i.i.d. random variables holds [32].

Fact 1. *For any set of discrete iid random variables $X_1, \ldots, X_n$, let $P(x_1, \ldots, x_n)$ denote $P(X_1 = x_1, \ldots, X_n = x_n)$. Given a permutation π mapping $\{1, 2, \ldots, n\}$ to itself,*

$$P(x_1, \ldots, x_n) = P(x_{\pi(1)}, \ldots, x_{\pi(n)}).$$

Thus a permuted set of query polynomials has the same marginal probability as an unpermuted set. However, if we just swap zero-polynomials into the unsynchronized indices, the probability of index i being desired conditioned on the query vector would be lower wherever a collusion of servers sees all zeros at index i. The client therefore adds a random, scalar-valued polynomial $p(X)$ with zero constant term to every entry of $\boldsymbol{q}(X)$. So the query vector gets transformed as follows:

$$\boldsymbol{q}(X) = \begin{bmatrix} q_1(X) \\ \ldots \\ q_n(X) \end{bmatrix} \implies \tilde{\boldsymbol{q}}(X) = \begin{bmatrix} q_1(X) + p(X) \\ \ldots \\ q_n(X) + p(X) \end{bmatrix},$$

where $\boldsymbol{q}(X)$ at unsynchronized indices is the zero polynomial. The added polynomial $p(X)$ acts like a one-time pad; it ensures that if a coalition of servers tries to break privacy, the unsynchronized index will not contain all zeros, but seemingly random numbers. If the databases were synchronized, there would be no zero-swapping, so this additional randomness would not be needed. The client then sends queries built from $\tilde{\boldsymbol{q}}(X)$; the ith server sends back the usual result, which by linearity is the same as $\boldsymbol{q}(X)^\intercal \boldsymbol{f}^{(i)} + p(X)\mathbf{1}^\intercal \boldsymbol{f}^{(i)}$. The server also returns the sum of all database records $T^{(i)} = \mathbf{1}^\intercal \boldsymbol{f}^{(i)}$, which can be precomputed. The client subtracts the noise $p(X)T^{(i)}$ from the ith server's reply, recovering the initially desired reply. Note that $p(X)$ need not be a polynomial;

the client could instead draw a random constant per server. We used random polynomials purely to simplify the notation.

Suppose a client wishes to retrieve records $f_{w_1}, \ldots, f_{w_\zeta}$. Phase 2 is described in Algorithm 6 and illustrated in Figure 3.8. Note that $\mathbf{A}$ changes between Algorithms 5 and 6, but serves the same purpose: to compress servers' responses. Setting $\mathbf{A}$ to the identity matrix in either algorithm is sufficient for success. For optimal compression though, the number of rows of $\mathbf{A}$ in Algorithm 5 should depend on the number of unsynchronized records, whereas in Algorithm 6, $\mathbf{A}$ needs only ζ rows—the number of desired records. This discrepancy arises because in phase 1, the client does not know the support of the sparse vector being measured (mis-synchronized indices), whereas in phase 2, it does (desired database indices). We can tolerate as many mis-synchronizations in Phase 2 as there are zero polynomials in the vector of random query polynomials. For a field size of 2^{ℓ_2} this amounts to $n/2^{\ell_2 \kappa}$ mis-synchronizations on average.

Theorem 3. *Suppose d databases contain at most $s = \gamma(n) \cdot n$ unsynchronized records, $0 \leq \gamma(n) \ll 1$. After running Algorithms 5 and 6, with matrix $\mathbf{A}$ in Algorithm 5 chosen as the parity check matrix of a systematic, MDS $(n + 2s, n, 2s + 1)$ error-correcting code, $\hat{\boldsymbol{r}}(0) = \boldsymbol{r}(0)$ with probability*

$$P(\textit{success}) \geq 1 - \underbrace{\gamma(n)}_{P(\textit{Error in Phase 1})} - \underbrace{e^{-2n(p-\gamma(n))^2}}_{P(\textit{Error in Phase 2})}.$$

where $p = 2^{-\ell_2 \cdot \kappa}$ is the probability of a random query polynomial being the zero polynomial, and $\gamma(n) \leq p$. If at most κ servers collude, the client's query is information-theoretically private (Proof in Appendix B.1).

The condition $\hat{\boldsymbol{r}}(0) = \boldsymbol{r}(0)$ means the client recovered all desired records by masking all unsynchronized database indices with zeros. In practice, if there are too few zero polynomials, the client can artificially introduce zero polynomials (which leaks some information) or construct a new query. Thus, the probability of successfully completing a PIR query is dominated by the ability to locate synchronization errors. If s is sublinear in n ($\gamma(n) \to 0$), this probability approaches 1 asymptotically. An analogous result holds for PULSE.

Efficiency: The computational and communication costs of this algorithm are listed in Table 3.2, using both the compression technique derived from [84] and Reed-Solomon codes [91]. We also compare our approach to a state-of-the-art scheme by Devet et al. [34] for processing concurrent queries. The comparison is unfair because [34] is not designed for unsynchronized databases; however, we do not know of any schemes that tackle our problem of interest. Table 3.2 suggests that **the required amount of communication and computation are nearly identical to state-of-the-art schemes when the number of unsynchronized records is small**.

The main communication overhead in our scheme compared to [34] is the downlink communication in Phase 1 when locating unsynchronized records ($2dsc \log_2 L$ bits). This overhead is asymptotically dominated by the uplink in Phase 2, so the asymptotic total communication complexity is the same between our scheme and [34]. Nonetheless, for practical database sizes, [34]

Protocol 6 A κ-private PIR algorithm for retrieving ζ records given the locations of s unsynchronized records.

Client:

1: Choose d random distinct indices $\alpha_1, \ldots, \alpha_d$ from $GF(2^{\ell_2})$. Define matrix $\mathbf{V}$, $V_{ij} = \alpha_{d-i+1}^{\kappa-j+1}$.
2: Pick vectors $\boldsymbol{a}_0, \ldots, \boldsymbol{a}_{\kappa-1}$, each element drawn uniformly at random from $GF(2^{\ell_2})$.
3: For each $u_i \in \{u_1, \ldots, u_s\}$, locate a distinct z_i such that $(\boldsymbol{a}_j)_{z_i} = 0$ for all $j \in [\kappa]$. If these z_i's do not exist, declare a failure and exit.
4: $\forall i \in [s]$ and $j \in [\kappa]$, swap $(\boldsymbol{a}_j)_{u_i}$ with $(\boldsymbol{a}_j)_{z_i}$.
5: Define query polynomial: $\boldsymbol{q}(X) = \boldsymbol{a}_0 X^\kappa + \boldsymbol{a}_1 X^{\kappa-1} + \ldots + \boldsymbol{a}_{\kappa-1} X + \sum_{i=1}^{\zeta} \boldsymbol{e}_{w_i}$.
6: Pick random polynomial $p(X)$ with $p(0) = 0$.
7: Send $\tilde{\boldsymbol{q}}(\alpha_i) = \boldsymbol{q}(\alpha_i) + p(\alpha_i)\mathbf{1}_n$ to server S_i.

Each honest-but-curious server (S_i, $i \in [d]$)**:**

8: Pick $\mathbf{A}$ as the first ζ rows of an $n \times n$ Vandermonde matrix. On receiving $\tilde{\boldsymbol{q}}(\alpha_i)$, compute $\tilde{\boldsymbol{r}}(\alpha_i) = \mathbf{A} \cdot (\tilde{\boldsymbol{q}}(\alpha_i) \otimes \boldsymbol{f}^{(i)})$.
9: Return $\tilde{\boldsymbol{r}}(\alpha_i)$ and $T^{(i)} = \sum_{j=1}^{n} f_j^{(i)}$ to the client.

Client:

10: From the server replies $\tilde{\boldsymbol{r}}(\alpha_i)$ and $T^{(i)}$, build matrix $\tilde{\mathbf{R}} = [\tilde{\boldsymbol{r}}(\alpha_1) - p(\alpha_1)T^{(1)} \ \ldots \ \tilde{\boldsymbol{r}}(\alpha_d) - p(\alpha_d)T^{(d)}]$.
11: Interpolate $\boldsymbol{y} = \tilde{\mathbf{R}} \cdot (\mathbf{V}^{-1})^\intercal \cdot [0 \ 0 \ \ldots \ 1]^\intercal$.
12: Define $\overline{\mathbf{V}}$ as columns $w_1, \ldots, w_\zeta$ of $\mathbf{A}$. Decode $\hat{\boldsymbol{r}}(0) = \overline{\mathbf{V}}^{-1}\boldsymbol{y}$. Return $\hat{\boldsymbol{r}}(0)$ (the desired records).
13: If $H(\hat{f}_w[\text{content}]) = \hat{f}_w[\text{hash}]$, the algorithm was successful. Otherwise, declare a failure.

has lower total communication. If the server precomputes hashes for the synchronization phase, the server's online computation cost is essentially equivalent between the two schemes. The client incurs additional computation to recover the locations of unsynchronized records and to interpolate the desired records.

Trading privacy for efficiency

Polynomial-swapping works when the database is large, but in a P2P network, servers may store smaller databases. Therefore, there may not be enough zero polynomials in practice to remove all the unsynchronized records from PIR queries. In this case, we can set polynomials to zero. This leaks information; we show that as long as the number of unsynchronized records grows sublinearly in the database size, the leaked information is asymptotically negligible, even with up to κ servers colluding.

Observation 1. *Suppose we build PIR queries that force unsynchronized query polynomials to be the zero polynomial. Suppose the client wishes to retrieve (synchronized) record f_w. Let s grow sublinearly in n, and suppose the true number of unsynchronized records among a set of databases s_o is distributed uniformly on the integers in interval $[0, s]$. Suppose at most κ servers are colluding.*

Algorithm	Communication	Server Computation	Client Computation
UPIR+PULSE, [84]	$d(\log d + 3.69sc \log L + \log_2 \zeta + \ell n + \zeta L)$	$nL(\zeta + 3.69sc)$	$O(ds) + O(\zeta^3 d^2 L)$
UPIR+RS, [91]	$d(\log d + 2sc \log L + \log \zeta + \ell n + \zeta L)$	$nL(\zeta + 2s)$	$O(dn^3) + O(\zeta^3 d^2 L)$ [56]
Goldberg, [51]	$dq\zeta(n + L)$	$2nL\zeta$	$O(\zeta dL)$

Table 3.2: Total communication (bits) and online computation (FLOPS) for unsynchronized PIR (UPIR) using PULSE and Reed-Solomon decoding, and a state-of-the-art collusion-resistant PIR scheme [51]. Parity symbols for the hashed database are precomputed. n = database size, d =number of servers, L =record size, ζ =number of records requested, 2^ℓ =field size (both phases).

Then

$$\lim_{n\to\infty} \frac{P(w = i | \boldsymbol{q}_i^{(1)} = 0, \ldots, \boldsymbol{q}_i^{(\kappa)} = 0)}{P(w = i |\, \boldsymbol{q}_i^{(1)}, \ldots, \boldsymbol{q}_i^{(\kappa)} \ \wedge\ \exists\, j \ s.t.\ \boldsymbol{q}_i^{(j)} \neq 0)} = 1$$

(Proof in Appendix B.1).

This says the likelihood of f_i being desired is nearly constant, whether the colluding servers observe zeros at index i or not; so zeroing out unsynchronized records reveals little information to a κ-coalition. For instance, if $n = 10,000$, field size is $GF(8)$, $s = 100$ unsynchronized records, and $\kappa = 3$ colluding servers, database indices with nonzero query values are almost 4x as likely to be desired as an index with all zeros. The remaining indices events have equal likelihood.

3.3 Experimental evaluation

To evaluate the practical performance of our scheme, we implemented it under the PERCY++ framework [34, 51], a C++ simulation of multi-server PIR algorithms.[8] Our metrics of interest were 1) probability of success, and 2) total query runtime, measured under realistic system settings. For these simulations, we used the PULSE compression approach, due to its simplicity and speed of decoding. Simulations were run on a virtual machine running on an Intel Core i7-620M processor, with one 2.67 GHz processing core and 1 GB of RAM. Measurements are averaged over 500 runs. Database records are 2048 bytes; all computations are over $GF(2^{16})$. We denote our scheme with 'UPIR' (unsynchronized PIR) in all figures and tables; PULSE indicates use of PULSE compression (correspondingly, RS for Reed-Solomon).

[8] PERCY++ code available at http://percy.sourceforge.net/; our modifications available at http://github.com/gfanti/P2P-PIR-Cpp.

Unsynchronized databases

Probability of Success

Certain patterns of query records and mis-synchronizations can lead to decoding errors, resulting in the client being not recovering the requested record. The probability of correctly retrieving the desired record increases as a function of communication cost.

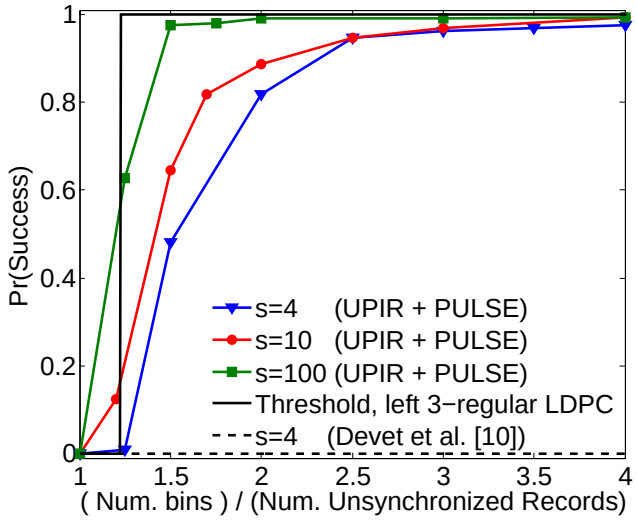


Figure 3.9: Probability of a successful PIR run as a function of the number of bins used in Phase 1 to identify the unsynchronized record indices. Each "bin" requires $3 \cdot L_H$ bits of downlink communication; we used $L_H = 48$ bits.

Figure 3.9 plots the probability of success as a function of the number of bins used for compression. Each 'bin' consists of three ratio measurements, each L_H bits. Even when the number of mis-synchronized records is small, this PIR scheme returns the desired record with high probability, as long as the servers use a small constant factor of the optimal number of bins. In this setup, we incur 7.2 kB extra communication to handle 100 mis-synchronized records. This is significant for our test database of 2 kB records. However, as records scale to larger sizes, e.g. a 3 order of magnitude increase, the overhead communication grows an order of magnitude to 72 kB—a fraction of the downlink communication required in Phase 2.

Runtime

The bottleneck in our scheme compared to the state-of-the-art [34] is locating unsynchronized records. In simulation, we did not pre-compute the synchronization replies for the servers, so these runtimes are worst-case estimates. Figure 3.10 shows runtime as a function of the number of unsynchronized database records. Our scheme runs within an order of magnitude of [34]'s runtime, and can complete a query in under 0.03 seconds in the face of 128 unsynchronized database

records. We ran [34] over synchronized databases, since it does not complete successfully when run over unsynchronized databases. Our runtime overhead is comparatively small, but it increases with the number of unsynchronized records.

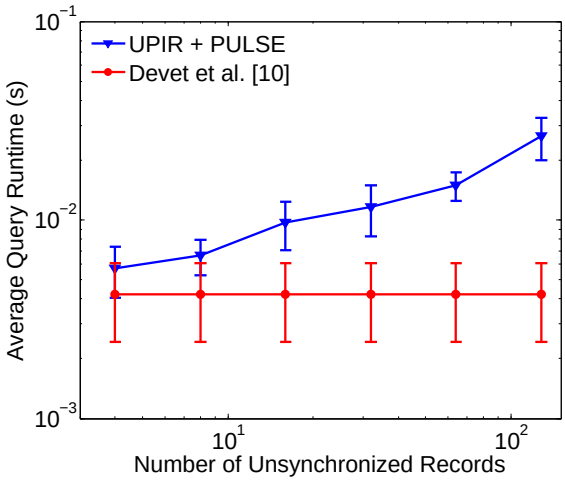


Figure 3.10: Average query runtime as a function of the number of unsynchronized records in the database. For our scheme, we used enough bins to guarantee a 0.95 probability of success.

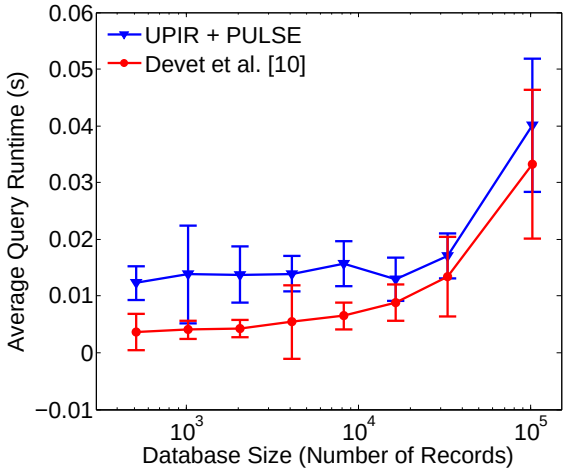


Figure 3.11: Query runtime as a function of database size, measured in number of records. We assume there are 32 unsynchronized records when running our scheme, and used 72 bins to locate the mis-synchronizations.

Figure 3.11 shows how runtime scales as a function of database size, measured in the number of database records. We fixed the number of unsynchronized records to 32, and used 72 bins to recover mis-synchronized records. This plot suggests that the runtime grows by less than an order of magnitude, even as the database size increases by several orders of magnitude. The overhead of our scheme compared to [34] does not increase as a function of database size, because the runtime overhead is dominated by locating the synchronization errors—a process that depends more heavily on the number of unsynchronized records than the database size. For a fixed number of mis-synchronizations, this overhead as a fraction of the total runtime actually decreases with database size. Thus we conclude that the raw number of unsynchronized records is the main contributing factor to added runtime.

3.4 Take-home Message

In this chapter, we presented the first multiserver PIR scheme to function when databases are not synchronized. The proposed scheme does not incur significant additional communication and

runtime costs when the number of unsynchronized records is small. When this number is small, the runtime increased by less than an order of magnitude in simulation.

These results suggest that in a networked PIR setting, the average fraction of unsynchronized records should be controlled to limit query times and overhead communication. Servers could update their records regularly to ensure that each server's database differs only slightly compared to the baseline.

This PIR scheme is a piece in the larger puzzle of distributed, privacy-preserving search systems. In particular, we will use the distributed source coding ideas from this chapter to construct efficient keyword search algorithms in Chapter 4.

Chapter 4

Efficient Private Search with Conjunctive Queries

In this chapter, we expand beyond the the data retrieval problem from Chapter 3 to consider the related problem of conjunctive keyword queries. This problem is important because search engine users typically process information by searching for keywords rather than retrieving data from pre-determined addresses. Keyword search alone is a more challenging problem than retrieval because it requires content-level analysis of data. As such, keyword search can incur a much higher computational and communication cost. Minimizing this overhead is a key objective.

When users submit keyword searches to search engines, those queries are typically *conjunctive*, or containing multiple keywords [66]. The implicit assumption is that the user wants results that feature *all* of those keywords. However, existing distributed private keyword search algorithms process such queries by returning a list of all documents that feature at least *one* of the queried keywords [24, 80]. This can lead to prohibitive downlink communication overhead, while also increasing the computational load of the client, which must sort through the results to retrieve the correct ones. For example, as of October 2015, Google reported 587 million results for the query "Edward OR Snowden", and only 33 million results for "Edward AND Snowden". In this example, a conjunctive query reduces the downlink communication cost by over an order of magnitude. More generally, conjunctive queries can significantly reduce downlink communication costs, compared to returning results that feature at least one queried keyword.

This claim clearly depends on the database and the kinds of queries people make; however, to back it up numerically, we examined the top 400 most popular queries on Google worldwide, measured on November 1, 2015 [105]. For each query, we measured the number of results for the query. If a query was conjunctive (e.g., $A \wedge B$), we also measured the number of results returned by the corresponding union query ($A \vee B$). We observed that 69 percent of queries had more than one keyword, and among those, 70 percent of queries had at least an order of magnitude more results for the $A \vee B$ query than the $A \wedge B$ query, and 33 percent had at least two orders of magnitude more

This chapter is based on joint work with Kannan Ramchandran.

results. While these observations are not necessarily representative of the databases that might be stored by a distributed, private search engine, they do suggest the importance of gracefully processing conjunctive queries.

A naive solution to the conjunctive private search problem is as follows: suppose the client wishes to query two keywords: "red cat". Then the server(s) could store an inverted database structure that is indexed by *pairs* of keywords (Figure 4.1). The client submits a normal PIR query for the wth index, where w maps to the desired pair of keywords (in Figure 4.1, "red cat" maps to index $w = 2$). Clearly, the downlink communication of this approach scales linearly with the number of conjunctive results, but the *storage* overhead grows as $(|\mathcal{K}|/m)^m$, where $|\mathcal{K}|$ is the number of keywords in the dictionary, and m is the number of queried keywords. Since the server should be prepared for queries of arbitrary length, the storage overhead alone becomes prohibitive. This is especially true in our distributed setting, where servers are not companies with dedicated infrastructure, but regular users who are contributing their computing resources to a privacy-preserving search engine. We must therefore explore alternative solutions, whose overhead does not grow so dramatically in the problem size.

$\mathcal{K}$ = {cat, dog, red}

(cat, dog): {f_1, f_2, f_4, …}
(cat, red): {f_1, f_3, …}
(dog, red): {f_2, f_4, f_6, …}

Figure 4.1: Example of an inverted database indexed by pairs of keywords. Each pair of keywords indexes a list of documents $\{f_i\}$ featuring that pair of keywords. $\mathcal{K}$ denotes the dictionary of keywords. Since $\mathcal{K}$ contains three keywords and the user is querying $m = 2$ keywords, the inverted structure has $\binom{3}{2}$ elements. $\binom{|\mathcal{K}|}{m}$ scales as $O(|\mathcal{K}|/m)^m$, which is prohibitive if a server were to accommodate conjunctive queries of arbitrary length.

Problem and Contributions

In our problem, a client wishes to conduct conjunctive keyword queries on a collection of documents, without revealing the query to the server. We wish to design a multi-server, private search scheme whose communication complexity depends only on the number of documents containing *all* of the queried keywords.

In this chapter, we present such an algorithm, which allows clients to privately search a collection of documents with conjunctive keyword queries without revealing the keywords to the server. In contrast to prior multi-server algorithms in this space, this algorithm's communication cost depends only on the number of documents that contain *all* the queried keywords. Critically, this algorithm does not incur any additional storage overhead compared to a single-keyword database search, and it can use the same data structure regardless of the number of keywords being queried.

Intuitively, the proposed algorithm relies on the fact that existing private *retrieval* algorithms are able to efficiently process queries for a union of keywords, returning documents that contain at least one query keyword. We exploit this to place a series of carefully-chosen union queries, and then process the results jointly in order to recover the relevant documents featuring the intersection of queried keywords. We analyze the performance of this technique theoretically, and demonstrate that it meets our efficiency constraints while returning provably correct results.

Related Work

The area of privacy-preserving keyword search is dominated by the study of queries over *encrypted data*. This area is primarily motivated by individuals or enterprises that wish to outsource private data on a third-party server without revealing the sensitive data to the storage server. The main idea is to encrypt the data with carefully-chosen cryptographic primitives and data structures that enable meaningful searches over encrypted data. Some work in this space focuses on the cryptographic primitives themselves [102, 18, 54]. Other work in this space instead focuses on building systems with existing cryptosystems. CryptDB, for instance, stores a database under multiple layers of encryption; when a client submits a query, the appropriate level of encryption is invoked that enables the most efficient processing possible [87]. Search over encrypted data is unrelated to our problem, because we are interested in privacy-preserving queries over *plaintext* data.

In the domain of privacy-preserving queries over plaintext data, researchers have studied private stream searches (PSS); in PSS, a client wishes to search a collection of documents (possibly streamed) and retrieve all documents that contain the queried keyword(s). As with PIR, there are both single-server and multi-server PSS schemes. The bulk of work in this space has focused on single-server PSS schemes, which rely on cryptographic assumptions [47, 83, 17, 45].[1] As a result, single-server PSS schemes can incur prohibitive computational costs; most research in this space has therefore focused on improving the efficiency of PSS, particularly by reducing the downlink communication cost [83, 17, 45]. As the name suggests, PSS is typically associated with streams of incoming data, rather than static databases. While this is not exactly our target application, PSS is the main existing technique for searching public data in a privacy-preserving way; in that sense, it is the most relevant prior work to our goals.

There has been comparatively little work on private keyword search over public data with multiple servers; indeed, the idea does not even have a unifying name in the literature. We therefore use "multi-server PSS" to describe a PSS algorithm that uses multiple servers with information-theoretic privacy guarantees, much like multi-server PIR. The idea was first introduced by Chor et al. [24]; in this work, the authors discuss methods for adapting existing PIR schemes to efficiently process keyword queries. Another similar paper by Goldberg et al. considers how to process complex SQL queries on public databases in a privacy-preserving manner [80]. Despite the strong efficiency gains associated with multi-server architectures, these works do not handle conjunctive queries explicitly. That is, if a client queries multiple keywords, existing PSS algorithms return documents that contain *at least one* of the queried keywords. This overhead unnecessary communi-

[1] Again, the exception to this rule is the trivial database transfer PSS scheme.

cation can be significant, particularly if the queried keywords are individually common. Moreover, existing approaches do not provide any mechanism for ranking private conjunctive queries on the basis of individual-keyword-based rankings. This is problematic because search engine rankings are critical for the usefulness of the search engine. Without rankings, it is difficult for clients to make sense of a potentially large set of results. Our work departs from prior art by developing multi-server PSS schemes that explicitly, efficiently deal with conjunctive queries and result ranking.

4.1 Setup and Notation

We use $[m]$ to denote the set $\{1, \ldots, m\}$. Boldface lowercase variables denote vectors ($\boldsymbol{a}$), and regular non-bolded variables denote scalars (a). Boldface uppercase variables denote matrices ($\mathbf{A}$), and uppercase non-bolded, subscripted variables denote matrix elements (A_{ij}). We use $\bigoplus$ and $\oplus$ to denote bitwise XOR (i.e., addition over a Galois field $GF(2^a)$), and we use $\sum$ and $+$ to denote addition over rings, such as integers modulo 2^a, which we denote by $\mathbb{Z}_{2^a}$.

Suppose there are d non-colluding, honest-but-curious servers. Each server stores an identical database of n ordered documents, $\boldsymbol{f} = [f_1, \ldots, f_n]$; for instance, a document might represent the contents of a webpage. This vector plays the same role as the database in Chapter 3, except now f_i represents a document rather than an indexed record in a cohesive database. There exists a global (ordered) dictionary of keywords $\mathcal{K} = \{k_1, \ldots, k_{|\mathcal{K}|}\}$, known to the client and all servers. Each document is associated with a subset of keywords from this dictionary; for instance, the ith document's subset, $K_i \subseteq \mathcal{K}$, might represent of all keywords in $\mathcal{K}$ that appear on webpage f_i.

A third party decides which keywords are significant in each document; the algorithm for deciding keyword prominence is beyond the scope of this paper, and we assume this mapping is given. The mapping is meant to represent a search engine indexing its search space according to some unknown, possibly proprietary algorithm. For example, importance might be tied to the number of times a keyword appears in a document.

Given such a database and dictionary, the client chooses to query a set of m keywords $X = \{x_1, \ldots, x_m\}$, $X \subseteq \mathcal{K}$. Let $Z = \{z_1, \ldots, z_s\}$ denote the indices of all documents in $\boldsymbol{f}$ that contain all the keywords specified by X. We assume that the number of desired documents s is much smaller than the number of total documents n. Based on X, the client generates a privacy-preserving query for each server, denoted $\boldsymbol{q}^{(t)}$, $t \in [d]$. Each server processes its query $\boldsymbol{q}^{(t)}$, and returns a reply $\boldsymbol{r}^{(t)}$ to the client. The client processes all the servers' replies to recover Z, the documents that feature *all* the keywords in X. Designing the algorithm that generates $\boldsymbol{q}^{(t)}$ and subsequently computes $\boldsymbol{r}^{(t)}$ as a function of $\boldsymbol{q}^{(t)}$ and the database is the objective of this paper.

Throughout this work, we use a running example based on a collection of documents illustrated in Figure 4.2. The dictionary in this example is $\mathcal{K} = \{\text{cat}, \text{dog}, \text{red}\}$. The number of documents is $n = 4$.

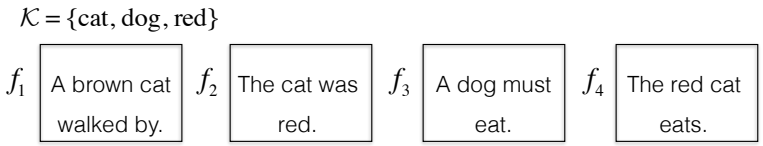


Figure 4.2: Documents used in running example. Each document features a different subset of dictionary keywords.

4.2 Background Concepts

Our algorithmic contributions combine ideas from the areas of private information retrieval, private stream search, and distributed source coding. We introduced the relevant background related to to private information retrieval and distributed source coding in Section 3.1, and therefore describe only PSS in greater detail here.

Private Stream Search

Conceptually, PIR is like browsing the Web while already knowing the URLs of the websites one wants to visit. A more realistic scenario is that a user knows what *content* she is looking for, but not the exact address or location of that content. Private stream search (PSS) addresses this challenge by allowing a user to learn if a queried keyword is present in a document, without revealing the keyword to the document server.

The most well-known algorithms in this space use a single-server architecture and rely on cryptographic assumptions [83, 17, 45]; these algorithms often utilize computationally-expensive cryptographic tools like homomorphic encryption.

We instead adopt a multi-server architecture similar to the one presented in [24]; effectively, we frame the problem as a multi-server PIR problem, and then modify the associated PIR algorithms. Posing PSS as a PIR problem is trivial. One option is for the tth server ($t \in [d]$) to store an inverted database, indexed by the keywords in $\mathcal{K}$. The ith database entry (corresponding to keyword k_i) contains a list of document IDs $\mathcal{S}_i$, such that for each $s \in \mathcal{S}_i$, f_s contains keyword k_i. The setup is illustrated in Fig. 4.3. Notice that unlike the naive solution proposed earlier, this setup only requires each server to store an inverted database indexed by each *single* keyword in the dictionary.

cat: {1, 2, 4}
dog: {3}
red: {2, 4}

Figure 4.3: The inverted database stored by each server. The list $\{1, 2, 4\}$ in the first row implies that f_1, f_2, and f_4 all feature the keyword "cat".

With this inverted database index, the client can simply submit a PIR query for the index of the desired keyword. For instance, if the client wanted keyword "cat", she would query record $w = 1$ in the example illustrated in Figure 4.3. Notice that the client must know the (ordered) keyword dictionary $\mathcal{K}$.

A major weakness of this approach arises when the client wishes to query multiple keywords. There do exist PIR schemes that allow the client to retrieve multiple records in one round of PIR [43, 59, 70], but in doing so, the client retrieves all tuples for documents that contain *at least one* desired keyword. The principal algorithmic challenge of our work is modifying these multi-server PIR algorithms so that the client can avoid wasting communication on documents that feature only a subset of desired keywords.

4.3 Algorithm Description

We tackle this problem in two phases, each requiring a full round of communication. During the first phase, the client learns which elements of the server's database meet its search criteria (i.e., which documents contain the desired keywords with the highest priority); in the second phase, the client retrieves those elements with a regular PIR query.

Phase 1: Find the desired documents

In this phase, the client issues a PIR-like query to learn which documents contain the desired keywords. Existing PIR techniques can reveal which documents contain keywords $k_{i_1} \vee \ldots \vee k_{i_m}$, where $\vee$ denotes logical OR, whereas we actually want the logical AND. At a high level, our approach is to compute carefully-chosen, correlated OR queries on different servers to recover the documents that contain $k_{i_1} \wedge \ldots \wedge k_{i_m}$, where $\wedge$ denotes the logical AND operation.

We present this algorithm by first stepping through a simple example. We then discuss how to generalize the example and give theoretical performance guarantees. We step through a private query containing $m = 2^c$ keywords for some positive integer c, and then generalize the algorithm to arbitrary numbers of keywords m; this process is outlined precisely in Algorithm 7, and the full pipeline is illustrated in Figure 4.5.

Client (Query Generation): Suppose the client wants records containing $m = 2$ keywords: $X = \{k_{i_1}, k_{i_2}\} = \{\text{"cat"}, \text{"red"}\}$. The client must contact $d = 2^m$ servers (in this case, $d = 4$), which are assumed to be non-colluding and honest-but-curious. The fact that this scheme requires an exponential number of servers in the query length is clearly not ideal; however, given that most search engine queries contain at most five keywords [66], this constraint is likely to be feasible in practice, particularly if each "server" is a node in a P2P network.

The client first generates a length-$|\mathcal{K}|$ noise vector $\boldsymbol{a}$, drawn i.i.d. from a Bernoulli(1/2) distribution. Each bit in $\boldsymbol{a}$ corresponds to one keyword in the dictionary. Now, the client uses $\boldsymbol{a}$ to generate four different query vectors: 1) $\boldsymbol{q}^{(1)} = \boldsymbol{a} + \mathbf{0}_{|\mathcal{K}|}$, where $\mathbf{0}_{|\mathcal{K}|}$ is the all-zero vector, 2) $\boldsymbol{q}^{(2)} = \boldsymbol{a} + \boldsymbol{e}_{i_1}$, where $\boldsymbol{e}_{i_1}$ is an indicator vector with a 1 at index i_1 and zeros elsewhere, 3)

Protocol 7 A multi-server, conjunctive query private search algorithm for $m = 2^c$ queries. Input: Queried keywords $X = \{k_{i_1}, \ldots, k_{i_m}\}$. Output: List of documents Z featuring keywords in X

Client:

Uniformly draw noise vector $\boldsymbol{a}$, $a_i \sim Bern(\frac{1}{2})$, $i \in [n]$
Generate the power set of X, $\mathcal{P}(X) = \{X_1, \ldots, X_{2^m}\}$
Define the ith query as $\boldsymbol{q}^{(i)} = \boldsymbol{a} + \boldsymbol{e}_{X_i}$
Send $\boldsymbol{q}^{(i)}$, $i \in [d]$, to server S_i

Each honest-but-curious server (S_i, $i \in [d]$)**:**

Initialize $\boldsymbol{r}^{(i)} = \mathbf{0}_n$
Let $a = c + 1$
for $j = 1$ to $|\mathcal{K}|$ **do**
    if $q_j^{(i)} = 1$ **then**
        for $s \in \mathcal{S}_j$ **do**
            $r_s^{(i)} = (r_s^{(i)} + 1) \mod 2^a$
Return $\boldsymbol{y}^{(i)} = \mathbf{A} \cdot \boldsymbol{r}^{(i)}$, computed over $GF(2^a)$

Client:

Compute $\boldsymbol{y} = \bigoplus_{j=1}^{d} \boldsymbol{y}^{(i)}$
Reconstruct $\hat{\boldsymbol{r}}$ from $\boldsymbol{y}$ and $\mathbf{A}$, e.g. using [84]
Return the nonzero indices of $\hat{\boldsymbol{r}}$

$\boldsymbol{q}^{(3)} = \boldsymbol{a} + \boldsymbol{e}_{i_2}$, and 4) $\boldsymbol{q}^{(4)} = \boldsymbol{a} + \boldsymbol{e}_{i_1,i_2}$, where $\boldsymbol{e}_{i_1,i_2}$ is 1 at both i_1 and i_2, and 0 elsewhere. Each of these four query vectors is sent to a different server. Notice that the indicator vectors added to the noise vector $\boldsymbol{a}$ collectively specify the *power set* of the queried keywords. That is, let $\mathcal{P}(X)$ denote the power set of X. Then the query vectors sent to each server are simply $\boldsymbol{a} + \boldsymbol{e}_{X'}$, for each $X' \subseteq \mathcal{P}(X)$. This is why we need $d = 2^m$ servers—because there are 2^m elements in $\mathcal{P}([m])$.

Server (Query Processing): Upon receiving a noisy query vector, the jth server generates a length-n list, initialized to $\mathbf{0}_n$, which we denote with $\boldsymbol{r}^{(j)}$. Each entry of $\boldsymbol{r}^{(j)}$ corresponds to a document. The server iterates over the $|\mathcal{K}|$ elements of the query vector $\boldsymbol{q}^{(j)}$. If the ith element of $\boldsymbol{q}^{(j)}$ is 0, the server does nothing; if the ith element of $\boldsymbol{q}^{(j)}$ is 1, the server iterates through the corresponding list of document indices in the ith element of the inverted database structure, $\mathcal{S}_i$. For each document index $s \in \mathcal{S}_i$, the server increments $\boldsymbol{r}_s^{(j)}$ by 1 over field $\mathbb{Z}_{2^a}$. At the end of this process, each server's $\boldsymbol{r}^{(j)}$ vector is a list detailing how many of the desired keywords appeared in each database document, modulo 2^a. The server's interpretation of a "desired keyword" is any keyword whose corresponding bit in $\boldsymbol{q}^{(j)}$ is set to 1. Because of the added noise $\boldsymbol{a}$, each server's set of "desired keywords" is very different from the true desired keywords. This procedure is illustrated for the non-private case when $\boldsymbol{a} = \mathbf{0}_{|\mathcal{K}|}$ in the top half of Figure 4.4.

Prior to returning the response vector $\boldsymbol{r}^{(i)}$ to the client, each server compresses $\boldsymbol{r}^{(i)}$. This step is omitted from Figure 4.4, but included in the pipeline diagram of Figure 4.5. The primary communication gains of our approach come from compression, which is only possible because the vectors $\boldsymbol{r}^{(i)}$ are additively sparse. This is not always guaranteed to be true, in which case we

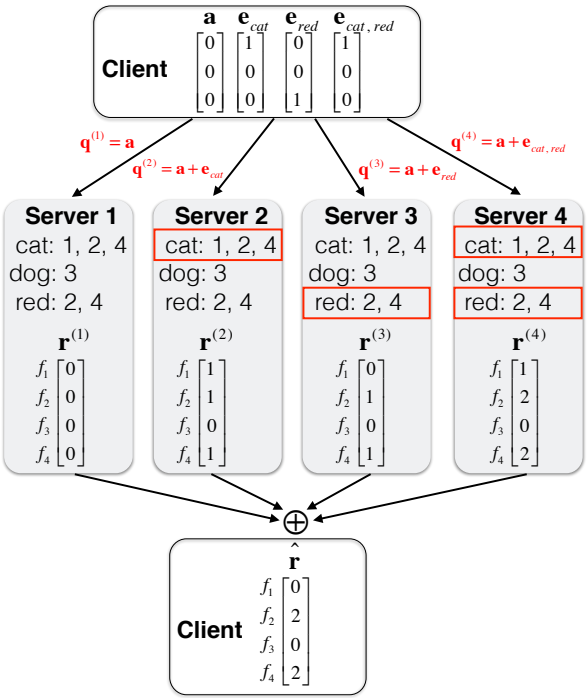


Figure 4.4: Additive sparsity arises across vectors. Here there is no added noise (i.e., $\boldsymbol{a} = \mathbf{0}_{|\mathcal{K}|}$). Red boxes indicate the (inverted) database entries requested from each server, and the final summation is taken over $GF(2^a)$ (i.e., a bitwise sum of the binary representation of each number in $\boldsymbol{r}^{(j)}$). Notice that in the summed vector, the only nonzero entries correspond to documents that contain all the queried keywords. In this example, $s = 2$. Because of this sparsity, each server can individually compress its (non-sparse) response vector $\boldsymbol{r}^{(j)}$ with a linear compression scheme to reduce the downlink communication to $O(s)$.

must alter Algorithm 7 slightly. However, when m is a power of two, this natural additive sparsity *always* holds, as demonstrated in the following proposition:

Proposition 4. *Suppose $m = 2^c$ for some $c > 0$, and suppose each server's reply vector $\boldsymbol{r}^{(t)}$, for $t \in [d]$, is computed according to Algorithm 7. Since the field requires at least $m + 1$ elements, we operate over fields $\mathbb{Z}_{2^a}$ and $GF(2^a)$ with $a = c + 1$. Then $\hat{r}_i \neq 0$ if and only if document f_i contains all m queried keywords.*

(Proof in Section C.1).

This proposition implies that under the specified conditions, the reply vectors from the servers are guaranteed to be additively sparse (i.e., $\hat{\boldsymbol{r}}$ is sparse), which in turn implies that they are compressible. By compressing $\boldsymbol{r}^{(i)}$ with a parity check matrix as described in Section 3.1, we can

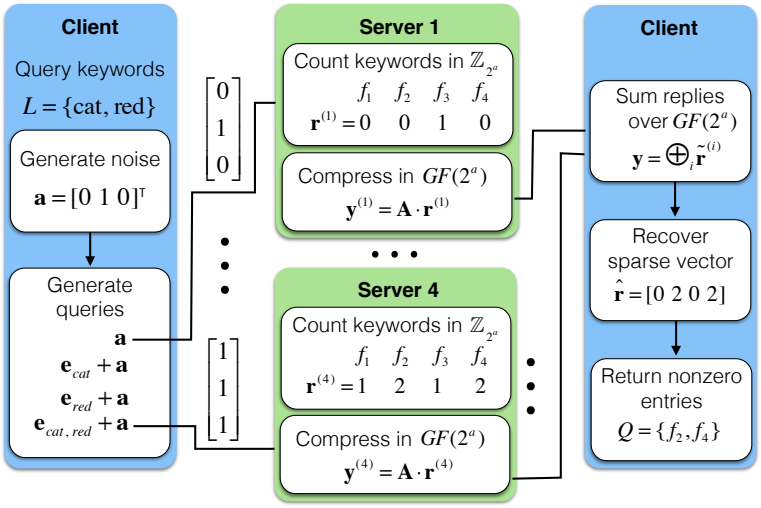


Figure 4.5: The processing pipeline for phase 1 when $m = 2^c$. Each client starts by generating 2^m binary query vectors, and sends one vector to each server. The 1's in each (noisy) query vector specify a subset of "requested" keywords. Each server counts how many of the requested keywords appear in each document. It then applies a post-processing function to that count, designed to enforce differential sparsity across servers. Finally, it compresses the resulting vector and returns it to the client, who decompresses and recovers the desired response.

ensure that with high probability (or deterministically, depending on the compression scheme), the client can recover the true $\hat{\boldsymbol{r}}$ from the compressed replies from all the servers. So each server can individually compute a compressed version of the post-processed reply vector, $\boldsymbol{y}^{(i)} = \mathbf{A} \cdot \boldsymbol{r}^{(i)}$, and return it to the client.

A note on field choice: Recall that servers compute their response vectors $\boldsymbol{r}^{(j)}$ by counting over field $\mathbb{Z}_{2^a}$. This field was chosen so that servers can distinguish between a keyword that appears two (or more) times and zero times—note that $1 + 1 = 0$ in $GF(2^a)$. However, we subsequently typecast these values into a Galois field for compression and reconstruction between servers because the sum of two binary vectors over such a field is the *binary* sum, or the XOR, of those vectors. The fact that XOR'ing is a linear operation over $GF(2^a)$ means that we can learn which documents contain *exactly one* of the desired keywords by simply summing $\boldsymbol{r}^{(2)} + \boldsymbol{r}^{(3)}$ over $GF(2^a)$. This in turn cancels out the interference in $\boldsymbol{r}^{(4)}$ from documents that contain only one desired keyword. These effects collectively allow us to construct the sparse vector $\hat{\boldsymbol{r}}$. So compression over $GF(2^a)$ has two main advantages: 1) it allows us to easily compute bitwise sums, which would be a nonlinear operation over $\mathbb{Z}_{2^a}$, and 2) it allows us to use distributed source coding to individually compress each server's non-sparse response $\hat{\boldsymbol{r}}$ without compromising the additive sparsity of the result.

Dealing with general m

The result in Proposition 4 is restrictive because it only holds for query lengths m that are a power of two. In principle, this could be handled by only submitting queries whose lengths are artificially padded to a power of two by inserting dummy keywords. However, doing so could significantly increase the communication overhead, since 2^m servers are queried. We improve the generality of this approach by introducing a *post-processing* function $h_m(\cdot)$, which is applied elementwise to each server's response vector prior to compression:

$$\tilde{\boldsymbol{r}}^{(i)} = h_m(\boldsymbol{r}^{(i)}).$$

In this more general case, with some abuse of notation, we redefine

$$\hat{\boldsymbol{r}} = \sum_{t=1}^{d} \tilde{\boldsymbol{r}}^{(t)},$$

so each server would send back $\mathbf{A}\cdot\tilde{\boldsymbol{r}}^{(i)}$ instead of $\mathbf{A}\cdot\boldsymbol{r}^{(i)}$ as before. This generalization is described in Appendix C.2, Algorithm 11. The question now becomes how to design $h_m(\cdot)$ so that $\hat{\boldsymbol{r}}$ has the desired sparsity pattern, regardless of the noise in the system. For example, when $m = 2^c$, we know that the *identity* post-processing function gives the desired result, so $h_m(j) = j$ for all $j \in GF(2^a)$; this function definition should be generalized for arbitrary $m \in \mathbb{N}_+$.

First, we introduce some background notation. Let $\boldsymbol{b}_m = (b_{a-1}, \ldots, b_0)$ denote the binary representation of the number of queried keywords m. For instance, if $m = 5$, we have $\boldsymbol{b} = (1\ 0\ 1)$. Let $\mathcal{Z}$ denote the set of bit indices in $\boldsymbol{b}$ that are *zero*, and $\mathcal{O}$ denote the set of bit indices in $\boldsymbol{b}$ that are *one*. In our example of $m = 5$, we have $\mathcal{Z} = \{1\}$ and $\mathcal{O} = \{0, 2\}$. Now define the sets

$$\mathcal{B} = \{\sum_{j\in B} 2^j,\ \forall B \in \mathcal{P}(\mathcal{O})\}$$

and

$$\overline{\mathcal{B}} = \{\sum_{j\in B} 2^j,\ \forall B \in \mathcal{P}(\mathcal{Z})\},$$

where $\sum_{j\in\emptyset} 2^j \equiv 0$. Intuitively, $\mathcal{B}$ contains all elements ℓ for which $\binom{m}{\ell}$ is odd, whereas $\overline{\mathcal{B}}$ describes the set of numbers that can be generated by summing *only* powers of two indexed by the zero-bits in m. In our example, we have $\mathcal{B} = \{0, 1, 4, 5\}$ and $\overline{\mathcal{B}} = \{0, 2\}$. Notice that $\mathcal{B} \cap \overline{\mathcal{B}} = \{0\}$, always. We use these sets to define a feasible choice of $h_m(\cdot)$, and subsequently prove its correctness.

Lemma 3. *Suppose each server's reply vector $\boldsymbol{r}^{(t)}$, for $t \in [d]$, is computed according to Algorithm 11. We choose the smallest a such that $m \in [2^a]$, and operate over fields $\mathbb{Z}_{2^a}$ and $GF(2^a)$. Then for any m and any choice of $h_m(\cdot)$, if f_i contains fewer than m queried keywords, then $\hat{r}_i = 0$. Additionally, if f_i contains exactly m queried keywords, then the following choice of $h_m(\cdot)$ ensures that $\hat{r}_i \neq 0$:*

$$h_m(i) = \begin{cases} 1 & \text{if } i \in \overline{\mathcal{B}} \\ 0 & \text{otherwise.} \end{cases} \tag{4.1}$$

(Proof in Section C.1).

This Lemma gives an achievable solution, but it is by no means necessary. Indeed, this solution does not reduce to the special case in Proposition 4 when m is a power of two.

Client (Result Processing): Once the client receives the reply vectors $\boldsymbol{y}^{(i)}$, it can compute $\boldsymbol{y} = \bigoplus_{i=1}^{d} \boldsymbol{y}^{(i)}$. By the linearity of the compression scheme, this is equal to $\mathbf{A}\hat{\boldsymbol{r}}$, the compressed version of $\hat{\boldsymbol{r}}$. The vector $\hat{\boldsymbol{r}}$ can then be recovered through the appropriate decoding algorithm for the parity check matrix $\mathbf{A}$. The output of this algorithm is a list of indices of the support of the reconstructed vector $\hat{\boldsymbol{r}}$.

Theorem 4.3.1. *Suppose Algorithm 11 is used to process a conjunctive query for keyword set X, under the conditions specified in Lemmas 4 and 3. The server and client both use a PULSE parity check matrix and corresponding decoding algorithm, as specified in [84]. Then for sufficiently large values of s (number of documents containing all the queried keywords) and n (number of documents in the collection), with probability at least $1 - O(s^{-3/2})$, the support of $\hat{\boldsymbol{r}}$ is equal to the set Z, which contains all documents containing all queried keywords. This output is obtained with $O(s)$ communication cost, while giving the client information-theoretic privacy guarantees.*

(Proof in Section C.1)

While the guarantees in [84] hold for "large enough" s and n, the PULSE compressive sensing construction empirically achieves good performance on small databases [84, 43]. This concludes the first round of communication, in which the client learns which documents contain the desired keywords.

Phase 2: Document retrieval

In this phase, the client retrieves the desired documents using a regular PIR query over the ordered list of documents $\boldsymbol{f} = [f_1, \ldots, f_n]$. This phase does not include any algorithmic innovations, but it is needed for completeness. If the client wishes to retrieve more than one of the documents reported in Phase 1, it can use a PIR scheme that enables the client to request multiple records at once, as in [43, 59, 70]. Otherwise, a normal PIR scheme suffices.

4.4 Performance

The proposed algorithm has downlink, per-server communication cost that scales linearly in s. However, a more meaningful metric of interest is the *total* downlink communication cost. In this

respect, the proposed algorithm is less appealing, since it uses 2^m servers. Table 4.1 quantifies the precise communication, computation, and storage cost of the proposed algorithm, compared to two naive baseline algorithms:

- *Naive scheme (1)*: Use an inverted database structure, indexed by each tuple of m keywords. If the maximum allowed number of keywords is $\overline{m}$, then with some abuse of notation, we let $|\mathcal{P}([\overline{m}])|$ denote the number of entries in this database: $|\mathcal{P}([\overline{m}])| = \sum_{i \in [\overline{m}]} \binom{|\mathcal{K}|}{i}$.[2] For a query set of keywords X, with $|X| \leq \overline{m}$, submit a PIR query for set X. This approach returns the exact conjunctive query results.

- *Naive scheme (2)*: Use an inverted database structure, indexed by each keyword individually (i.e., the database has $|\mathcal{K}|$ entries). Submit one PIR query for each queried keyword, or use a scheme like [43] that allows multiple concurrent queries. This approach returns all documents that contain at least one queried keyword, and the client must thereafter filter out the documents indexed by Z. This approach is representative of the current state-of-the-art, modulo the underlying PIR scheme [24, 80].

Table 4.1 assumes the PIR scheme from [23], but it could be modified to reflect an underlying PIR scheme of choice. The factor of $\log(n)$ in the downlink communication of the proposed scheme stems from the compression field size, which must contain at least n elements. This is a straw-man comparison because neither of these approaches are proposed in the literature to deal specifically with conjunctive queries. However, since this particular problem has not been studied before, we can only compare against straw-man schemes.

The benefits of the proposed scheme may not be obvious from the asymptotic costs. First, notice that the storage cost of Naive Scheme (1) scales exponentially faster than the other two schemes due to the $|\mathcal{P}([\overline{m}])|$ term, so we do not view it as a practically-viable solution for a distributed system. Given this, the real advantage of the proposed scheme is that the downlink communication scales according to s instead of $m\hat{s}$. The difference between variables s and $\hat{s}$ depends heavily on the dataset in question. However, we have observed empirically that these constants differ significantly for popular queries in search engines like Google—often by an order of magnitude or more.

4.5 Take-home Message

In this chapter, we have introduced an algorithm for conducting multi-server, privacy-preserving keyword queries on a collection of plaintext documents. The proposed construction enables conjunctive queries for any arbitrary number of keywords. Moreover, we have proved that this algorithm—unlike prior art in this area—incurs a downlink communication cost that scales linearly in the number of documents containing the *conjunction* of queried keywords, rather than the *union*.

[2] This is an abuse of notation because the summation is a partial enumeration of the power set of $[\mathcal{K}]$, whereas previously we used $\mathcal{P}(\cdot)$ to denote the full power set of the argument.

Algorithm	Communication		Server Computation	Storage per Server
	Uplink	Downlink		
Proposed	$2^m\|\mathcal{K}\| + 2n$	$3.69s \cdot 2^m \log(n) + 2L$	$O(sn\|\mathcal{K}\| + L)$	$O(n(L + \|\mathcal{K}\|))$
Naive (1)	$2\|\mathcal{P}([\overline{m}])\|$	$2\hat{s}L$	$O(\|\mathcal{P}([\overline{m}])\| \cdot \hat{s}L)$	$O(n(L + \|\mathcal{P}([\overline{m}])\|))$
Naive (2)	$2m\|\mathcal{K}\|$	$2m\hat{s}L$	$O(m\|\mathcal{K}\|\hat{s}L)$	$O(n(L + \|\mathcal{K}\|))$

Table 4.1: Total uplink and downlink communication (bits), server-side online computation (FLOPS), and server storage. n = database size, $\mathcal{K}$ = keyword dictionary, X = document size, s = number of documents featuring all m queried keywords, $\hat{s}$ = number of documents featuring a single given keyword, maximized over all keywords in the dictionary. Yellow cells denote the portion of each algorithm with the most overhead compared to the other baseline algorithms.

While this algorithm answers a previously unanswered problem in a theoretical sense, its practical implications may be limited. This is because although each server *individually* incurs a small communication cost, the number of servers scales exponentially in the number of keywords. This magnifies the total communication cost by a constant, but potentially large, factor. As such, the proposed algorithm in its current state may not result in significant communication savings unless the intersection of query terms is significantly smaller than the union of query terms (e.g., by an order of magnitude or more). While we have empirically observed that this is true for many queries, it may not hold for general datasets and queries. Additionally, the incurred computation cost of this algorithm is slightly higher—by a constant factor—than the corresponding naive solution that relies exclusively on existing PIR techniques. Therefore, our proposed algorithm may not be efficient enough for adoption in practical settings. However, we maintain that the problem is an important one, and our hope is that the proposed distributed source coding approach can inspire more efficient and practical solutions.

Chapter 5

Future Work and Conclusions

In this thesis, we have introduced algorithms enabling anonymous messaging and privacy-preserving search. This section includes a discussion of important future work, as well as primary conclusions based on our results.

5.1 Anonymous Messaging

In the area of anonymous messaging, we introduced adaptive diffusion. Adaptive diffusion is a spreading protocol that protects against deanonymization of authors in messaging networks by means of a message's spreading pattern over a graph. We demonstrated that over regular trees, adaptive diffusion is optimal under the snapshot adversarial model and asymptotically optimal in the tree degree under the spy-based adversarial model. We also demonstrated empirically that adaptive diffusion performs nearly optimally over real social networks that are finite and cyclic.

This algorithm is lightweight and efficient enough to scale to large social networks. Moreover, the anonymity properties are strong enough—even over realistic networks—to give significantly better anonymity than existing networks. However, if such an algorithm were to make its way into a practical system, there are a few important research questions that need to be answered beforehand:

- Provide relay nodes with deniable plausibility. Our work so far is focused on protecting the author of a source. However, in practice, a user might be implicated simply by virtue of forwarding a message, even if the user was not the true author. Therefore, it is important to introduce a deniability mechanism that allows relays to claim that they did not actively approve a message. This may be feasible through using a randomized response mechanism, providing relays with differential privacy guarantees on the binary random variable of whether or not a user 'liked' a given message.

- Integrate mechanisms defending against cyberbullying and spamming. Cyberbullying is a major problem in anonymous messaging social networks [26]. However, anonymity is at odds with accountability. If a powerful user, or a large coalition of users can deanonymize

a message to identify cyberbullies or spammers, then a powerful adversary can bypass the protections introduced by our system design. As such, it may be interesting to investigate methods that allow a coalition of users to *prevent* a message from spreading further, without revealing the author of the message. This could be tackled through systems-level innovations, for instance by making use of a semi-centralized architecture that allows users to report objectionable content.

5.2 Private Search

In the area of private search, we introduced two new algorithmic innovations. The first is a multi-server private information retrieval (PIR) scheme that is correct with high probability even when the distributed servers have outdated views of the database replicates. This problem is important in the context of distributed, peer-to-peer (P2P) networks, where databases are not necessarily managed and maintained with the consistency of centralized services. The second algorithm enables privacy-preserving conjunctive keyword queries over a public collection of documents, with a communication cost that scales linearly in the number of documents featuring the intersection of queried keywords.

Our overarching goal for this line of work is a distributed, privacy-preserving search engine for medium-sized collections of data (e.g., the Snowden leaked documents, popular news outlets). Clearly, such a search engine would operate on keyword queries, most of which would presumably be conjunctive (assuming user query patterns match those of existing search engines [66]). The algorithms we have proposed in this work are a step in the direction of this broader goal; however, they may not be efficient enough in their current state to build a usable, scalable system.

In order to bridge this gap, we propose a few follow-up research questions:

- Reduce the number of servers needed in our conjunctive query keyword search algorithm. Ideally, this number would scale at most linearly in the number of keywords queried. In our existing formulation, sparsity comes naturally (without any postprocessing) because undesired documents get summed over a Galois field an even number of times. It may be possible to achieve a similar condition by simply reframing the optimization over fewer servers, and including some postprocessing in order to enforce the desired sparsity condition.

- Make the protocol collusion-resistant. The protocol we propose is resistant to an honest-but-curious adversary of non-colluding servers. However, in a P2P network, servers may collude in order to break users' privacy. There is a great deal of prior work on collusion-resistant PIR [12, 51, 34]. However, these techniques cannot be applied to the private keyword search algorithm in Chapter 4, because the existing schemes rely on the linearity of the underlying PIR scheme. Our keyword search, on the other hand, includes nonlinearities that prevent these techniques from generalizing. The intersection function is itself nonlinear, so we believe this problem should be tackled by developing nonlinear collusion-resistant algorithms, rather than attempting to find a linear conjunctive keyword search algorithm.

5.3 Final Thoughts

The questions considered in this thesis aim to empower people to use information-sharing networks without facing retribution for the content they share or consume. In principle, the privacy-preserving messaging and search ideas presented here could be combined into one integrated network that provides privacy-preserving information access *and* dissemination. However, we believe both problems are significant of their own right, and there is a public need for services that satisfy each privacy constraint individually. For instance, we have discussed an anonymous messaging social network, and a P2P, privacy-preserving search engine. While this thesis tackles a few algorithmic challenges in the space, there are still significant hurdles to bringing these ideas to the public in a practical sense.

Bibliography

[1] FireChat. https://opengarden.com/firechat.

[2] Secret. https://www.secret.ly.

[3] Whisper. http://whisper.sh.

[4] Yik Yak. http://www.yikyakapp.com.

[5] B. J. Alge. Effects of computer surveillance on perceptions of privacy and procedural justice. *J*ournal of Applied Psychology, 86(4):797, 2001.

[6] Andris Ambainis. Upper bound on the communication complexity of private information retrieval. In *A*utomata, Languages and Programming, pages 401–407. Springer, 1997.

[7] M. Barbaro and T. Zeller. A face is exposed for AOL searcher no. 4417749. *N*ew York Times, August 2006. http://www.nytimes.com/2006/08/09/technology/09aol.html.

[8] Kevin Bauer, Damon McCoy, Ben Greenstein, Dirk Grunwald, and Douglas Sicker. Physical layer attacks on unlinkability in wireless lans. In *P*rivacy Enhancing Technologies, pages 108–127. Springer, 2009.

[9] E. Baumer, P. Adams, V. D. Khovanskaya, T. C. Liao, M. E. Smith, V. Schwanda Sosik, and K. Williams. Limiting, leaving, and (re) lapsing: an exploration of facebook non-use practices and experiences. In *S*IGCHI, pages 3257–3266. ACM, 2013.

[10] A. Beimel, Y. Ishai, E. Kushilevitz, and T. Malkin. One-way functions are essential for single-server private information retrieval. In *S*ymposium on theory of computing, pages 89–98. ACM, 1999.

[11] A. Beimel, Y. Ishai, and T. Malkin. Reducing the servers' computation in private information retrieval: PIR with preprocessing. *J*. Cryptology, 17(2):125–151, 2004.

[12] A. Beimel and Y. Stahl. Robust information-theoretic private information retrieval. In *S*ecurity in Communication Networks, pages 326–341. Springer, 2003.

[13] Amos Beimel, Yuval Ishai, and Eyal Kushilevitz. General constructions for information-theoretic private information retrieval. *J*ournal of Computer and System Sciences, 71(2):213–247, 2005.

[14] Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and J-F Raymond. Breaking the o (n 1 (2k-1)/) barrier for information-theoretic private information retrieval. In *F*oundations of Computer Science, 2002. Proceedings. The 43rd Annual IEEE Symposium on, pages 261–270. IEEE, 2002.

[15] M. Ben-Or, S. Goldwasser, and A. Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *P*roc. of STOC, pages 1–10. ACM, 1988.

[16] A. C. Berry. The accuracy of the gaussian approximation to the sum of independent variates. *T*ransactions of the american mathematical society, 49(1):122–136, 1941.

[17] J. Bethencourt, D. Song, and B. Waters. New constructions and practical applications for private stream searching. In *I*EEE Symp. on Security and Privacy, pages 134 –139, 2006.

[18] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *E*urocrypt, pages 506–522. Springer, 2004.

[19] N. Borisov, G. Danezis, and I. Goldberg. DP5: A private presence service. *P*roceedings on Privacy Enhancing Technologies, 2015(2):1–21, 2015.

[20] J.L. Carter and M.N. Wegman. Universal classes of hash functions. In *P*roc. of STOC, pages 106–112. ACM, 1977.

[21] D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Communications of the ACM*, 24(2):84–90, 1981.

[22] D. L. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *J*ournal of cryptology, 1(1):65–75, 1988.

[23] B. Chor, O. Goldreich, E. Kushilevitz, and M. Sudan. Private information retrieval. In *P*roc. of IEEE FOCS, pages 41–50, Milwaukee, WI, 1995.

[24] Benny Chor, Niv Gilboa, and Moni Naor. *P*rivate information retrieval by keywords. Citeseer, 1997.

[25] A. Clauset, C. R. Shalizi, and M. E. J. Newman. Power-law distributions in empirical data. *S*IAM Review, 51(4):661–703, 2009.

[26] W. Cohen. In rise of Yik Yak, profits and ethics collide. *N*ew York Times, 2015.

[27] H. Corrigan-Gibbs, D. Boneh, and D. Mazières. Riposte: An anonymous messaging system handling millions of users. In *S*ymposium on Security and Privacy. IEEE, 2015.

[28] H. Corrigan-Gibbs and B. Ford. Dissent: accountable anonymous group messaging. In *CCS*. ACM, 2010.

[29] T. M. Cover and J. A. Thomas. *E*lements of information theory. John Wiley & Sons, 2012.

[30] B. Danev, H. Luecken, S. Capkun, and K. El Defrawy. Attacks on physical-layer identification. In *C*onference on Wireless network security, pages 89–98. ACM, 2010.

[31] G. Danezis and P. Mittal. Sybilinfer: Detecting sybil nodes using social networks. In *N*DSS. San Diego, CA, 2009.

[32] B. De Finetti. Probability, induction, and statistics. 1972.

[33] C. Devet and I. Goldberg. The best of both worlds: Combining information-theoretic and computational PIR for communication efficiency. Technical report, University of Waterloo.

[34] C. Devet, I. Goldberg, and N. Heninger. Optimally robust private information retrieval. *I*ACR Cryptology ePrint Archive, 2012:83, 2012.

[35] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. Technical report, DTIC Document, 2004.

[36] W. Dong, W. Zhang, and C. W. Tan. Rooting out the rumor culprit from suspects. In *ISIT*, pages 2671–2675. IEEE, 2013.

[37] Z. Dvir and S. Gopi. 2-server pir with sub-polynomial communication. *a*rXiv preprint arXiv:1407.6692, 2014.

[38] K. Efremenko. 3-query locally decodable codes of subexponential length. *J*ournal on Computing, 41(6):1694–1703, 2012.

[39] Larry M Elison and Dennis NettikSimmons. Right of privacy. *M*ont. L. Rev., 48:1, 1987.

[40] G. Fanti, P. Kairouz, S. Oh, K. Ramchandran, and P. Viswanath. Hiding the rumor source. *a*rxiv preprint arxiv:1509.02849, 2015.

[41] G. Fanti, P. Kairouz, S. Oh, and P. Viswanath. Spy vs. spy: Rumor source obfuscation. In *S*IGMETRICS/PERFORMANCE. ACM, 2015.

[42] G. Fanti and K. Ramchandran. Multi-server private information retrieval over unsynchronized databases. In *A*llerton, 2014.

[43] G. Fanti and K. Ramchandran. Efficient private information retrieval over unsynchronized databases. *J*ournal of Selected Topics in Signal Processing, 2015.

[44] A. Fazeli, A. Vardy, and E. Yaakobi. Pir with low storage overhead: Coding instead of replication. *a*rXiv preprint arXiv:1505.06241, 2015.

[45] M. Finiasz and K. Ramchandran. Private stream search at the same communication cost as a regular search: Role of ldpc codes. In *P*roc. IEEE Int. Symposium on Information Theory, pages 2556–2560, 2012.

[46] V. Fioriti and M. Chinnici. Predicting the sources of an outbreak with a spectral technique. *a*rXiv preprint arXiv:1211.2333, 2012.

[47] M. J. Freedman, Y. Ishai, B. Pinkas, and O. Reingold. Keyword search and oblivious pseudorandom functions. In *T*heory of Cryptography, pages 303–324. Springer, 2005.

[48] Michael J Freedman and Robert Morris. Tarzan: A peer-to-peer anonymizing network layer. In *Proceedings of ACM CCS*, 2002.

[49] Craig Gentry and Zulfikar Ramzan. Single-database private information retrieval with constant communication rate. In *A*utomata, Languages and Programming, pages 803–815. Springer, 2005.

[50] S. Goel, M. Robson, M. Polte, and E. Sirer. Herbivore: A scalable and efficient protocol for anonymous communication. Technical report, Cornell University, 2003.

[51] I. Goldberg. Improving the robustness of private information retrieval. In *I*EEE Symp. on Security and Privacy, pages 131–148. IEEE, 2007.

[52] O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious rams. *J*ournal of the ACM (JACM), 43(3):431–473, 1996.

[53] P. Golle and A. Juels. Dining cryptographers revisited. In *E*urocrypt, pages 456–473. Springer, 2004.

[54] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *CCS*, pages 89–98. ACM, 2006.

[55] D. Gross. Yahoo hacked, 450,000 passwords posted online. *CNN Tech*, July 13 2012. http://www.cnn.com/2012/07/12/tech/web/yahoo-users-hacked.

[56] F. G. Gustavson. Analysis of the berlekamp-massey linear feedback shift-register synthesis algorithm. *I*BM Journal of Research and Development, 20(3):204–212, 1976.

[57] T. E. Harris. *T*he theory of branching processes. Courier Corporation, 2002.

[58] D. C. Howe and H. Nissenbaum. Trackmenot: Resisting surveillance in web search. *L*essons from the Identity Trail: Anonymity, Privacy, and Identity in a Networked Society, 23:417–436, 2009.

[59] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Batch codes and their applications. In *S*ymposium on Theory of computing, pages 262–271. ACM, 2004.

[60] M. Jawad, P. Serrano-Alvarado, and P. Valduriez. Protecting data privacy in structured P2P networks. In *D*ata Management in Grid and Peer-to-Peer Systems. Springer, 2009.

[61] N.L. Johnson and S. Kotz. *U*rn models and their application. Wiley New York, 1977.

[62] C. Karlof and D. Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *A*d hoc networks, 1(2):293–315, 2003.

[63] J.O. Koehler. *STASI: The untold story of the East German secret police*. Basic Books, 1999.

[64] M. Kolountzakis. The study of translational tiling with fourier analysis. In *F*ourier analysis and convexity, pages 131–187. Springer, 2004.

[65] J. Korner and K. Marton. How to encode the modulo-two sum of binary sources (corresp.). *T*ransactions on Information Theory, 25(2):219–221, 1979.

[66] F. Lardinois. Hitwise: Search queries are getting longer. *r*eadwrite, 2009.

[67] P. Lewis and D. Rushe. Revealed: How Whisper app tracks 'anonymous' users. *T*he Guardian, 2014.

[68] A. Y. Lokhov, M. Mézard, H. Ohta, and L. Zdeborová. Inferring the origin of an epidemic with dynamic message-passing algorithm. *a*rXiv preprint arXiv:1303.5315, 2013.

[69] M. Luby, M. Mitzenmacher, M. Amin Shokrollahi, and D. Spielman. Efficient erasure correcting codes. *I*nformation Theory, IEEE Transactions on, 47(2):569–584, 2001.

[70] Wouter Lueks and Ian Goldberg. Sublinear scaling for multi-client private information retrieval. In *F*inancial Cryptography and Data Security, 2015.

[71] W. Luo, W. P. Tay, and M. Leng. Identifying infection sources and regions in large networks. *T*ransactions on Signal Processing, 61(11):2850–2865, 2013.

[72] D. Lyon. Surveillance, snowden, and big data: Capacities, consequences, critique. *B*ig Data & Society, 1(2):2053951714541861, 2014.

[73] D. McCullagh. Verizon draws fire for monitoring app usage, browsing habits. *CNET*, October 16 2012. Retrieved from http://news.cnet.com/8301-13578_3-57533001-38/verizon-draws-fire-for-monitoring-app-usage-browsing-habits/.

[74] E. A. Meirom, C. Milling, C. Caramanis, S. Mannor, A. Orda, and S. Shakkottai. Localized epidemic detection in networks with overwhelming noise. 2014.

[75] C. Milling, C. Caramanis, S. Mannor, and S. Shakkottai. Network forensics: Random infection vs spreading epidemic. In *S*IGMETRICS. ACM, 2012.

[76] C. Milling, C. Caramanis, S. Mannor, and S. Shakkottai. On identifying the causative network of an epidemic. In *A*llerton Conference, pages 909–914, 2012.

[77] C. Milling, C. Caramanis, S. Mannor, and S. Shakkottai. Detecting epidemics using highly noisy data. In *M*obiHoc, pages 177–186, 2013.

[78] M. Mitzenmacher and G. Varghese. Biff (bloom filter) codes: Fast error correction for large data sets. In *I*nformation Theory Proceedings (ISIT), 2012 IEEE International Symposium on, pages 483–487. IEEE, 2012.

[79] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *SODA*, pages 448–457. SIAM, 2001.

[80] Femi Olumofin and Ian Goldberg. Privacy-preserving queries over relational databases. In *P*rivacy enhancing technologies, pages 75–92. Springer, 2010.

[81] F.G. Olumofin. *P*ractical Private Information Retrieval. PhD thesis, University of Waterloo, August 2011.

[82] R. Ostrovsky and W. E. Skeith III. A survey of single-database private information retrieval: Techniques and applications. In *P*ublic Key Cryptography, pages 393–411. Springer, 2007.

[83] R. Ostrovsky, W. Skeith, and O. Patashnik. Private searching on streaming data. *J*ournal of Cryptology, 20:397–430, 2007.

[84] S. Pawar. PULSE: Peeling-based ultra-low complexity algorithms for sparse signal estimation. *P*h.D. Thesis, 2013. http://www.eecs.berkeley.edu/Pubs/TechRpts/2013/EECS-2013-215.pdf.

[85] B. Pinkas and T. Reinman. Oblivious ram revisited. In *A*dvances in Cryptology, pages 502–519. Springer, 2010.

[86] P. C. Pinto, P. Thiran, and M. Vetterli. Locating the source of diffusion in large-scale networks. *P*hysical review letters, 109(6):068702, 2012.

[87] R. A. Popa, C. Redfield, N. Zeldovich, and H. Balakrishnan. Cryptdb: protecting confidentiality with encrypted query processing. In *SOSP*, pages 85–100. ACM, 2011.

[88] S.S. Pradhan and K. Ramchandran. Distributed source coding using syndromes. *T*ransactions on Information Theory, 49(3):626–643, 2003.

[89] B. A. Prakash, J. Vreeken, and C. Faloutsos. Spotting culprits in epidemics: How many and which ones? In *I*CDM, volume 12, pages 11–20, 2012.

[90] M. Rabin. How to exchange secrets with oblivious transfer. *IACR* Cryptology ePrint Archive, 2005:187, 2005.

[91] I. S. Reed and G. Solomon. Polynomial codes over certain finite fields. *J*SIAM, 8(2):300–304, 1960.

[92] M. K. Reiter and A. Rubin. Anonymous web transactions with crowds. *C*ommunications of the ACM, 42(2):32–48, 1999.

[93] D. Shah and T. Zaman. Finding rumor sources on random graphs. *a*rXiv preprint arXiv:1110.6230, 2011.

[94] D. Shah and T. Zaman. Rumors in a network: Who's the culprit? *I*nformation Theory, IEEE Transactions on, 57(8):5163–5181, Aug 2011.

[95] Nihar B Shah, KV Rashmi, and Kannan Ramchandran. One extra bit of download ensures perfectly private information retrieval. In *I*nformation Theory (ISIT), 2014 IEEE International Symposium on, pages 856–860. IEEE, 2014.

[96] A. Shamir. How to share a secret. *C*ommunications of the ACM, 22(11):612–613, 1979.

[97] E. Shi, T-H. H. Chan, E. Stefanov, and M. Li. Oblivious ram with o ((logn) 3) worst-case cost. In *A*SIACRYPT, pages 197–214. Springer, 2011.

[98] A. Shontell. 7 people who were arrested because of something they wrote on facebook. *B*usiness Insider, 2013. Retrieved from http://www.businessinsider.com/people-arrested-for-facebook-posts-2013-7#ixzz3iBbceA3w.

[99] R. Singel. Point, click ... eavesdrop: How the FBI wiretap net operates. *W*ired, 2007.

[100] R. Sion and B. Carbunar. On the computational practicality of private information retrieval. In *P*roc. of NDSS, pages 2006–2016, 2007.

[101] A. Smith. 6 new facts about Facebook. *P*ew Research, 2014.

[102] D. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *S*ymposium on Security and Privacy, pages 44–55. IEEE, 2000.

[103] Foreign Staff. British woman 'sentenced to 20 years in iran for facebook posts'. *T*he Telegraph, May 2014.

[104] E. Stefanov, E. Shi, and D. Song. Towards practical oblivious ram. *a*rXiv preprint arXiv:1106.3652, 2011.

[105] Google Trends. https://www.google.com/trends/.

[106] J. Ugander, B. Karrer, L. Backstrom, and C. Marlow. The anatomy of the Facebook social graph. *a*rXiv preprint arXiv:1111.4503, 2011.

[107] B. Viswanath, A. Mislove, M. Cha, and K. P. Gummadi. On the evolution of user interaction in facebook. In *W*orkshop on Online social networks, pages 37–42. ACM, 2009.

[108] M. Waidner and B. Pfitzmann et al. The dining cryptographers in the disco: Unconditional sender and recipient untraceability with computationally secure serviceability. *EUROCRYPT*, 89:690, 1989.

[109] Z. Wang, W. Dong, W. Zhang, and C. W. Tan. Rumor source detection with multiple observations: Fundamental limits and algorithms. 2014.

[110] S. Wehner and R. De Wolf. Improved lower bounds for locally decodable codes and private information retrieval. In *A*utomata, Languages and Programming, pages 1424–1436. Springer, 2005.

[111] P. Winter and S. Lindskog. How the great firewall of china is blocking Tor. *F*OCI, 2012.

[112] D. I. Wolinsky, H. Corrigan-Gibbs, B. Ford, and Aaron A. Johnson. Dissent in numbers: Making strong anonymity scale. In *O*SDI, pages 179–182. USENIX, 2012.

[113] Y.-C.Chang. Single database private information retrieval with logarithmic communication. In *I*nformation Security and Privacy, pages 50–61. Springer, 2004.

[114] E. Yang, J. Xu, and K. Bennett. Private information retrieval in the presence of malicious failures. In *P*roc. of COMPSAC, pages 805–810, 2002.

[115] Sergey Yekhanin. Towards 3-query locally decodable codes of subexponential length. *J*ournal of the ACM (JACM), 55(1):1, 2008.

[116] Sergey Yekhanin. Private information retrieval. *C*ommunications of the ACM, 53(4):68–73, 2010.

[117] K. Zhu, Z. Chen, and L. Ying. Locating contagion sources in networks with partial timestamps. *a*rXiv preprint arXiv:1412.4141, 2014.

[118] K. Zhu and L. Ying. A robust information source estimator with sparse observations. *a*rXiv preprint arXiv:1309.4846, 2013.

Appendix A

Proofs from Chapter 2

Proof of Theorem 4.3.1

Spreading rate. Under Protocol 1, G_T is a complete $(d-1)$-ary tree (with the exception that the root has d children) of depth $T/2$ whenever T is even. Whenever T is odd, with probability $\alpha_d(T,h)$, G_T is again such a $(d-1)$-ary tree of depth $(T+1)/2$. With probability $1-\alpha_d(T,h)$, G_T is made up of two $(d-1)$-ary trees of depth $(T-1)/2$ each with their roots connected by an edge. Therefore, it follows that when $d>2$, N_T is given by

$$N_T \;=\; \begin{cases} 1, & T=0, \\ \frac{2(d-1)^{(T+1)/2}}{d-2}-\frac{2}{d-2}, & T\geq 1,\ T \text{ odd, w.p. } (1-\alpha)\ , \\ \frac{d(d-1)^{(T+1)/2}}{d-2}-\frac{2}{d-2}, & T\geq 1,\ T \text{ odd, w.p. } \alpha\ , \\ \frac{d(d-1)^{T/2}}{d-2}-\frac{2}{d-2}, & T\geq 2,\ T \text{ even}\ ; \end{cases} \tag{A.1}$$

Similarly, when $d=2$, N_T can be expressed as follows:

$$N_T \;=\; \begin{cases} 1, & T=0, \\ T+1, & T\geq 1,\ T \text{ odd, w.p. } (1-\alpha)\ , \\ T+2, & T\geq 1,\ T \text{ odd, w.p. } \alpha\ , \\ T+2, & T\geq 2,\ T \text{ even}\ ; \end{cases} \tag{A.2}$$

The lower bound on N_T in Equation (2.6) follows immediately from the above expressions.

Probability of detection. For any given infected graph G_T, the virtual source v_T cannot have been the source node, since the true source always passes the token at timestep $t=1$. So $\mathbb{P}(G_T|v=v_T)=0$. We claim that for any two nodes that are not the virtual source at time T, $u,w\in G_T$, $\mathbb{P}(G_T|u)=\mathbb{P}(G_T|w)>0$. This is true iff for any non-virtual-source node v, there exists a sequence of virtual sources $v_i{}_{i=0}^{T}$ that evolves according to Protocol 1 with $v_0=v$ that results in the observed G_T, and for all $u,w\in G_T\setminus\{v_T\}$, this sequence has the same likelihood. In a tree, a unique path exists between any pair of nodes, so we can always find a valid path of virtual sources from a candidate node $u\in G_T\setminus\{v_T\}$ to v_T. We claim that any such path leads

to the formation of the observed G_T. Due to regularity of G and the symmetry in G_T, for even T, $\mathbb{P}(G_T|v^{(1)}) = \mathbb{P}(G_T|v^{(2)})$ for all $v^{(1)}, v^{(2)} \in G_T$ with $\delta_H(v^{(1)}, v_T) = \delta_H(v^{(2)}, v_T)$. Moreover, recall that the $\alpha_d(t,h)$'s were designed to satisfy the distribution in Equation (2.4). Combining these two observations with the fact that we have $(d-1)^h$ infected nodes h-hops away from the virtual source, we get that for all $v^{(1)}, v^{(2)} \in G_T \setminus \{v_T\}$, $\mathbb{P}(G_T|v^{(1)}) = \mathbb{P}(G_T|v^{(2)})$. For odd T, if the virtual source remains the virtual source, then G_T stays symmetric about v_T, in which case the same result holds. If the virtual source passes the token, then G_T is perfectly symmetric about the edge connecting v_{T-1} and v_T. Since both nodes are virtual sources (former and present, respectively) and $T > 1$, the adversary can infer that neither node was the true source. Since the two connected subtrees are symmetric and each node within a subtree has the same likelihood of being the source by construction (Equation (2.4)), we get that for all $v^{(1)}, v^{(2)} \in G_T \setminus \{v_T, v_{T-1}\}$, $\mathbb{P}(G_T|v^{(1)}) = \mathbb{P}(G_T|v^{(2)})$. Thus at odd timesteps, $\mathbb{P}(\hat{v}_{ML} = v^*) \geq 1/(N_T - 2)$.

Proof of Proposition 2.2.1

First, under Protocol 1 (adaptive diffusion) with $\alpha_d(t,h) = 0$, G_T is a complete $(d-1)$-ary tree (with the exception that the root has d children) of depth $T/2$ whenever T is even. G_T is made up of two complete $(d-1)$-ary trees of depth $(T-1)/2$ each with their roots connected by an edge whenever T is odd. Therefore, it follows that N_T is a deterministic function of T and is given by

$$N_T = \begin{cases} 1, & T = 0, \\ \frac{2(d-1)^{(T+1)/2}}{d-2} - \frac{2}{d-2}, & T \geq 1,\ T \text{ odd}, \\ \frac{d(d-1)^{T/2}}{d-2} - \frac{2}{d-2}, & T \geq 2,\ T \text{ even}; \end{cases} \tag{A.3}$$

The lower bound on N_T in Equation (2.9) follows immediately from the above expression.

For any given infected graph G_T, it can be verified that any non-leaf node could not have generated G_T under the Tree Protocol. In other words, $\mathbb{P}(G_T|v \text{ non-leaf node}) = 0$ and v could not have started the rumor. On the other hand, we claim that for any two leaf nodes $v_1, v_2 \in G_T$, we have that $\mathbb{P}(G_T|v_1) = \mathbb{P}(G_T|v_2) > 0$. This is true because for each leaf node $v \in G_T$, there exists a sequence of state values $\{s_{1,u}, s_{2,u}\}_{u \in G_T}$ that evolves according to the Tree Protocol with $s_{1,v} = 1$ and $s_{2,v} = 0$. Further, the regularity of the underlying graph G ensures that all these sequences are equally likely. Therefore, the probability of correct rumor source detection under the maximum likelihood algorithm is given by $\mathbb{P}_{ML}(T) = 1/N_{l,T}$, where $N_{l,T}$ represents the number of leaf nodes in G_T. It can be also shown that $N_{l,T}$ and N_T are related to each other by the following expression

$$N_{l,T} = \frac{(d-2)N_T + 2}{d-1}. \tag{A.4}$$

This proves the expression for $\mathbb{P}\big(\hat{v}_{ML} = v^*\big)$ given in (2.10).

Expected distance. For any $v^* \in G$ and any T, $\mathbb{E}[\delta_H(v^*, \hat{v}_{ML})]$ is given by

$$\mathbb{E}[\delta_H(v^*, \hat{v}_{ML})] = \sum_{v \in G} \sum_{G_T} \mathbb{P}(G_T|v^*)\mathbb{P}(\hat{v}_{ML} = v)\delta_H(v^*, v). \tag{A.5}$$

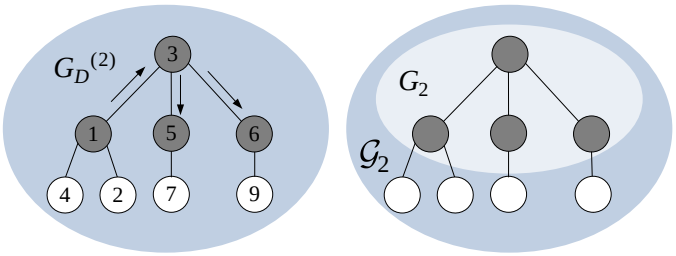


Figure A.1: One realization of the random, irregular-tree branching process. Although each realization of the random process $G_D^{(t)}$ yields a labelled graph, the adversary observes G_T and $\mathcal{G}_T$, which are unlabelled. White nodes are uninfected, grey nodes are infected.

As indicated above, no matter where the rumor starts from, G_T is a $(d-1)$-ary tree (with the exception that the root has d children) of depth $T/2$ whenever T is even. Moreover, $\hat{v}_{ML} = v$ with probability $1/N_{l,T}$ for all v leaf nodes in G_T. Therefore, the above equation can be solved exactly to obtain the expression provided in the statement of the proposition.

Proof of Theorem 2.2.2

We first analyze the probability of detection for any given estimator (see Eq. (A.10)); we then show that the estimator in (2.16) is a MAP estimator, maximizing this probability of detection. Finally, we show that using the MAP estimator in 2.16 gives the probability of detection in Eq. (2.17).

We begin with some definitions. Consider the following random process, in which we fix a source v^* and generate a (random) labelled tree $G_D^{(t)}$ for each time t and for a given degree distribution D. At time $t = 0$, $G_D^{(t)}$ consists of a single node v^*, which is given a label 1. The source v^* draws a degree d_1 from D, and generates d_1 child nodes, labelled in order of creation (i.e., 2 through $d_1 + 1$). At the next time step, $t = 1$, the source picks one of these neighbors uniformly at random to be the new virtual source and infects that neighbor. According to Protocol 1, each time a node v is infected, v draws its degree d_v from D, then generates $d_v - 1$ labelled child nodes. So at the end of time $t = 1$, $G_D^{(1)}$ contains the source and its uninfected neighbors, as well as the new virtual source and its uninfected neighbors. An example of $G_D^{(2)}$ is shown in Figure A.1 (left panel) with $d_1 = 3$ and virtual source at node 3. Grey nodes are infected and white nodes are uninfected neighbors. Note that the node labelled 1 is always exactly one hop from a leaf of $G_D^{(t)}$ for all $t > 0$; also, nodes infect their neighbors in ascending order of their labels. The leaves of $G_D^{(t)}$ represent the uninfected neighbors of infected leaves in standard adaptive diffusion spreading over a given graph. Define $\Omega_{(t,D)}$ as the set of all labelled trees generated at time t according to this random process.

At some time T, the adversary observes the snapshot of infected subgraph G_T. Notice that we

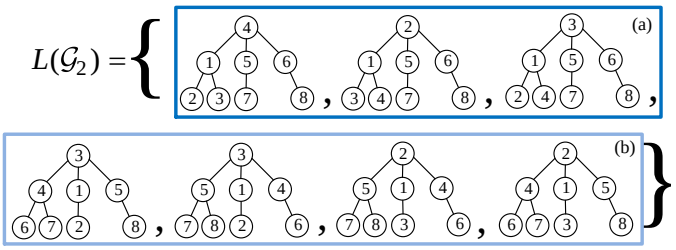


Figure A.2: $L(\mathcal{G}_2)$ for the snapshot $\mathcal{G}_2$ illustrated in Figure A.1. Boxes (a) and (b) illustrate the two families partitioning $L(\mathcal{G}_2)$.

do not need to generate the entire contact network, since G_T is conditionally independent of the rest of the contact network given its one-hop neighbors. Hence, the we only need to generate (and consider) the one hop neighbors of G_T at any given T. We use $\mathcal{G}_T$ to denote this random graph that includes G_T and its one hop neighbors as generated according to the previously explained random process. Notice that the adversary only observes $\mathcal{G}$, which is an unlabelled snapshot of the infection and its one hop neighbors (see Figure A.1, right panel). We refer to the leaves of G_T as 'infected leaves', denoted by ∂G_T, and the leaves of $\mathcal{G}_T$ as 'uninfected leaves' denoted by $\partial \mathcal{G}_T$. Define

$$L(\mathcal{G}_T) \equiv \{\tilde{G} \in \Omega_{(T,D)} \mid U(\tilde{G}) = \mathcal{G}_T\},$$

i.e., the set of all labelled graphs (generated according to the described random process) whose unlabelled representation $U(\tilde{G})$ is equal to the snapshot $\mathcal{G}_T$. Figure A.2 illustrates $L(\mathcal{G}_T)$ for the graph $\mathcal{G}_2$ in Figure A.1.

We define a family $C_{\mathcal{G}_T,v} \subseteq L(\mathcal{G}_T)$ as the set of all labelled graphs whose labeling could have been generated by breadth-first labeling of $\mathcal{G}_T$ starting at node $v \in \partial G_T$. Here breadth-first labeling is a valid order of traversal for a breadth-first search of $\mathcal{G}_T$ starting at node v. We restrict v to be a valid source for an adaptive diffusion spread—that is, it is an infected leaf in ∂G_T. Note that a BFS labeling starting from two different nodes on the unlabelled tree can yield the same labelled graph. In Figure A.2, boxes (a) and (b) illustrate the two families contained in $L(\mathcal{G}_2)$.

Let $\mathbb{P}(C_{\mathcal{G},v}) \equiv \mathbb{P}(G_D^{(T)} \in C_{\mathcal{G},v})$ denote the probability that the labelled graph $G_D^{(T)}$ whose snapshot is $\mathcal{G}$ is generated from a node v. From the definition of the random process for generating labelled graphs, we get

$$\mathbb{P}(C_{\mathcal{G}_T,v}) = \underbrace{\left(\prod_{w \in G_T} \mathbb{P}_D(d_w)\right)}_{\text{degrees of } G} \underbrace{Q(\mathcal{G}_T, v)}_{\text{virtual sources}} \underbrace{|C_{\mathcal{G}_T,v}|}_{\substack{\text{count of} \\ \text{isomorphisms}}} \tag{A.6}$$

where $\mathbb{P}_D(d)$ is the probability of observing degree d under degree distribution D, and

$$Q(\mathcal{G}_T, v) = \frac{\mathbb{1}_{v \in \partial G_T}}{d_v \prod_{w \in \Phi_{v,v_T} \setminus \{v,v_T\}} (d_w - 1)}$$

is the probability of passing the virtual source from v to the virtual source v_T given the structure of $\mathcal{G}_T$, where Φ_{v,v_T} is the unique path from v to v_T in $\mathcal{G}_T$. Eq. (A.6) holds because for all instances in $C_{\mathcal{G}_T,v}$, the probability of the degrees of the nodes and the probability of the path of the virtual source remain the same.

The probability of observing a given snapshot $\mathcal{G}_T$ is precisely $\mathbb{P}(G_D^{(T)} \in L(\mathcal{G}_T))$. Notice that $C_{\mathcal{G}_T,v}$ partitions $L(\mathcal{G}_T)$ in to family of labelled trees that are generated from the same source. This give the following decomposition:

$$\mathbb{P}(G_D^{(T)} \in L(G_T)) = \sum_{v \in \mathcal{C}_{\mathcal{G}_T}} \mathbb{P}(C_{\mathcal{G}_T,v}), \tag{A.7}$$

where we define $\mathcal{C}_{\mathcal{G}_T}$ as the set of possible candidates of the source that generate distinct labelled trees, i.e.

$$\mathcal{C}_{\mathcal{G}_T} \equiv \{v \in G_T \,|\, C_{\mathcal{G}_T,v} \neq C_{\mathcal{G}_T,v'} \; \forall \, v' \in \mathcal{C}_{\mathcal{G}_T},\; v' \neq v\} \,. \tag{A.8}$$

Notice that this set is not unique, since there can be multiple nodes that represent the same family $C_{\mathcal{G}_T,v}$. We pick one of such node v to represent the class of nodes that can generate the same family of labelled trees. We use this v to index these families and not to denote any particular node in ∂G_T.

Consider an estimate of the source $\hat{v}(\mathcal{G}_T)$. In general, $\hat{v}(\mathcal{G}_T)$ is a random variable, potentially selected from a set of candidates. We define detection ($\overline{D}$) as the event in which $\hat{v}(\mathcal{G}_T) = v_1(G_D^{(T)})$; i.e., the estimator outputs the node that started the random process. We can partition the set of candidate nodes ∂G_T, by grouping together those nodes that are indistinguishable to the estimator into classes. Precisely, we define a subset of nodes indexed by $v \in \mathcal{C}_{\mathcal{G}_T}$,

$$\chi_{\mathcal{G}_T,v} \equiv \{v' \in \partial G_T \,|\, C_{\mathcal{G}_T,v} = C_{\mathcal{G}_T,v'}\} \,. \tag{A.9}$$

For a given snapshot, there are as many classes as there are families. In Figure A.2, the class associated with family (a) has one element—namely, the node labeled '1' in family (a). The class associated with family (b) contains two nodes: the node labeled '1' in family (b), and the node labeled '5' in the rightmost graph of family (b), since both nodes give rise to the same family.

We consider, without loss of generality, an estimator that selects a node in a given class with probability $\mathbb{P}(\hat{v}(\mathcal{G}_T) \in \chi_{\mathcal{G}_T,v})$. Notice that $|\chi_{\mathcal{G}_T,v}|$ denotes the number of (indistinguishable) source candidates in this class. From Eq. (A.7), the probability of detection given a snapshot is

$$\mathbb{P}(\overline{D}|\mathcal{G}_T) = \frac{\mathbb{P}\left(G_D^{(T)} \in L(\mathcal{G}_T) \wedge \overline{D}\right)}{\mathbb{P}(G_D^{(T)} \in L(\mathcal{G}_T))}. \tag{A.10}$$

$$= \frac{\sum\limits_{v \in \mathcal{C}_{\mathcal{G}_T}} \mathbb{P}(C_{\mathcal{G}_T,v}) \mathbb{P}\left(\overline{D} \,\middle|\, G_D^{(T)} \in C_{\mathcal{G}_T,v}\right)}{\sum_{v \in \mathcal{C}_{\mathcal{G}_T}} \mathbb{P}(C_{\mathcal{G}_T,v})} \tag{A.11}$$

where $\mathbb{P}(\overline{D}|G_D^{(T)} \in C_{\mathcal{G}_T,v}) = \mathbb{P}(\hat{v}(\mathcal{G}_T) \in \chi_{\mathcal{G}_T,v})/|\chi_{\mathcal{G}_T,v}|$. We use the following observation:

Lemma 4.

$$\frac{\mathbb{P}(C_{\mathcal{G}_T,v})/|\chi_{\mathcal{G}_T,v}|}{\sum_{v\in\mathcal{C}_{\mathcal{G}_T}}\mathbb{P}(C_{\mathcal{G}_T,v})} = \frac{1}{d_{v_T}\prod\limits_{\substack{w\in\phi(v,v_T)\\ \setminus\{v,v_T\}}}(d_w-1)}. \tag{A.12}$$

(Proof in Section A)
Substituting Equation (A.12) into Equation (A.11), we get that

$$\mathbb{P}(\overline{D}|\mathcal{G}_T) \;=\; \sum_{v\in\mathcal{C}_{\mathcal{G}_T}} \frac{\mathbb{P}(\hat{v}(\mathcal{G}_T)\in\chi_{\mathcal{G}_T,v})}{d_{v_T}\prod\limits_{\substack{w\in\phi(v,v_T)\setminus\\ \{v,v_T\}}}(d_w-1)}.$$

Since

$$\frac{1}{d_{v_T}\prod\limits_{\substack{w\in\phi(v,v_T)\setminus\\ \{v,v_T\}}}(d_w-1)} \le \frac{1}{\min\limits_{v\in\mathcal{C}_{\mathcal{G}_T}} d_{v_T}\prod\limits_{\substack{w\in\phi(v,v_T)\\ \setminus\{v,v_T\}}}(d_w-1)},$$

and $\sum_{v\in\mathcal{C}_{\mathcal{G}_T}}\mathbb{P}(\hat{v}(\mathcal{G}_T)\in\chi_{\mathcal{G}_T,v})=1$, it must hold that

$$\mathbb{P}(\overline{D}|\mathcal{G}_T) \;\le\; \frac{1}{\min\limits_{v\in\mathcal{C}_{\mathcal{G}_T}} d_{v_T}\prod\limits_{\substack{w\in\phi(v,v_T)\\ \setminus\{v,v_T\}}}(d_w-1)}.$$

This upper bound on the detection probability is achieved exactly if we choose weight $\mathbb{P}(\hat{v}(\mathcal{G}_T)\in\chi_{\mathcal{G}_T,v})=1$ for the class(es) minimizing the product $\prod_{w\in\phi(v,v_T)\setminus\{v,v_T\}}(d_w-1)$, i.e.,

$$\hat{v}(G_T) = \arg\min_{v\in\partial G_T}\prod_{\substack{w\in\phi(v,v_T)\\ \setminus\{v,v_T\}}}(d_w-1).$$

Proof of Lemma 4

We have that

$$\mathbb{P}(C_{\mathcal{G}_T,v}) = \underbrace{\left(\prod_{w\in G_T}\mathbb{P}_D(d_w)\right)}_{\text{degrees of } G}\underbrace{Q(\mathcal{G}_T,v)}_{\text{virtual sources}}\underbrace{|C_{\mathcal{G}_T,v}|}_{\substack{\text{count of}\\ \text{isomorphisms}}}$$

where v is a feasible source for the adaptive diffusion process, i.e., a leaf of the infection G_T.
The proof of the lemma proceeds in four steps:

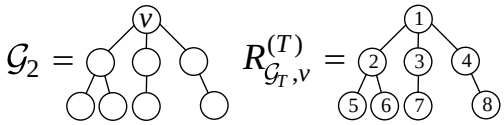


Figure A.3: A realization of the random labeling process given an unlabeled snapshot.

1. We first recursively define a function $H(\mathcal{G}_T, v)$ that is equal to $|C_{\mathcal{G}_T,v}|$. This function is defined over any balanced, undirected tree and node; the tree need not be generated via the previously-described adaptive diffusion branching process. In addition to $H(\mathcal{G}_T, v)$, we are interested in $H(\mathcal{G}_T, v_T)$.

2. We show that

$$\mathbb{P}(C_{\mathcal{G}_T,v}) = \left(\prod_{v\in G_T} \mathbb{P}_D(d_v)\right) H(\mathcal{G}_T, v_T) \times \frac{|\chi_{\mathcal{G}_T,v}|}{d_{v_T} \prod\limits_{\substack{w\in\phi(v,v_T)\\ \setminus\{v,v_T\}}} (d_w - 1)}. \tag{A.13}$$

3. We show that

$$\sum_{v\in\mathcal{C}_{\mathcal{G}_T,v}} \mathbb{P}(C_{\mathcal{G}_T,v}) = \left(\prod_{v\in G_T} \mathbb{P}_D(d_v)\right) H(\mathcal{G}_T, v_T). \tag{A.14}$$

4. We combine steps (2) and (3) to show the result.

Step 1 We wish to define $H(\mathcal{G}_T, v)$—a function that counts the number of distinct, isomorphic graphs generated by a breadth-first search of a balanced tree $\mathcal{G}_T$, rooted at node v. Consider a random process defined as follows. Given $\mathcal{G}_T$ and root node v, the process starts at v and labels it 1. For each neighbor w of node 1, the process randomly orders w's unlabelled neighbors, and labels them in order of traversal. The process proceeds to label nodes in a breadth-first fashion, traversing each node's unlabelled neighbors in a randomly-selected order, until all nodes have been visited. Let $R^{(t)}_{\mathcal{G}_T,v}$ denote a labelled tree generated according to the described random process (see Figure A.3).

The function $H(\mathcal{G}_T, v)$ counts the number of distinct graphs that can result from this random process over $\mathcal{G}_T$ when starting from node v. More precisely, define $\mathcal{R}^{(T)}_{\mathcal{G}_T,v}$ as the set of all possible trees $R^{(T)}_{\mathcal{G}_T,v}$ generated according to this random labeling. $H(\mathcal{G}_T, v)$ is defined as the size of $\mathcal{R}^{(T)}_{\mathcal{G}_T,v}$. Figure A.4 illustrates $\mathcal{R}^{(T)}_{\mathcal{G}_T,v}$ for $\mathcal{G}_T$ and v shown in Figure A.3. In that example, $H(\mathcal{G}_T, v) = 3$.

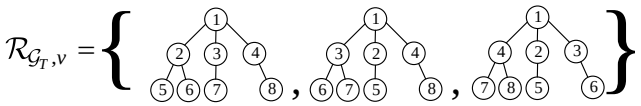
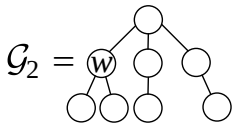


Figure A.4: The set $\mathcal{R}^{(T)}_{\mathcal{G}_T,v}$ for the snapshot and node specified in Figure A.3.

Recall that $\mathcal{G}_T$ is a balanced tree. The Jordan center of this tree is denoted by v_T. If $\mathcal{G}_T$ was generated according to adaptive diffusion, v_T would be the virtual source at time T. Although we say $\mathcal{G}_T$ is rooted at v, we define each node's children with respect to v_T. That is, node z is among w's children if z is a neighbor of w and $z \notin \phi(w, v_T)$.

Let $\mathcal{G}_T^{v_i \to v_j}$ denote the subtree of $\mathcal{G}_T$ rooted at node v_j with node v_i as parent of v_j (let $\mathcal{G}_T^{v_1 \to v_1} = \mathcal{G}_T$). Each node v_i in $\mathcal{G}_T$ will have some number of child subtrees. Some of these subtrees may be identical (i.e., given a realization $R_{\mathcal{G}_T,v}$ of the labeling random process, they would be isomorphic); let k_v denote the number of distinguishable subtrees of node v. We use $\Delta_1^v, \ldots, \Delta_{k_v}^v$ to denote the number of each distinct subtree appearing among the child subtrees of node v (recall children are defined with respect to v_T). For example, node v in graph $\mathcal{G}_T$ in Figure A.3 (left panel) has $\Delta_1^v = 1$ and $\Delta_2^v = 2$, since the first of v's child subtrees is equal only to itself, and the second (middle) subtree is isomorphic to the subtree on the right. If there exists a neighboring, unvisited subtree rooted at a parent of v, then we say $\Delta_0^v = 1$ (by definition, there will only be one such subtree, and it cannot be equal to any child subtrees because G_T is balanced). Otherwise, we say $\Delta_0^v = 0$. This distinction becomes relevant if $v \neq v_T$. For example the figure below shows a tree that is rooted at $w \neq v_T$. In computing $H(\mathcal{G}_T, w)$, we have $\Delta_0^w = 1$ because there is an unvisited branch from w that contains v_T, and $\Delta_1^w = 2$ because both child subtrees of w are identical.



Let γ^v denote the unvisited neighbors of node v in $\mathcal{G}_T$. We give a recursive expression for computing $H(\mathcal{G}_T, v)$.

Lemma 5.

$$H(\mathcal{G}_T, v) = \binom{d_v}{\Delta_0^v, \Delta_1^v, \ldots, \Delta_k^v} \prod_{w \in \gamma^v} H(\mathcal{G}_T^{v \to w}, w). \tag{A.15}$$

Proof. We show this by induction on the depth λ of $\mathcal{G}_T$ (rooted at v). For $\lambda = 1$, $\mathcal{G}_T$ has a node v and d_v neighbors. Every realization of the random breadth-first labeling of $\mathcal{G}_T$ will yield an identical graph since the neighbors of v are indistinguishable, so $H(\mathcal{G}_T, v) = \binom{d_v}{d_v} = 1$.

Now suppose Equation (A.15) holds for all graph-node pairs (G_T, v) with $\lambda < \lambda_o$; we want to show that it holds for $\lambda = \lambda_o$. We can represent $\mathcal{G}_T$ as a root node v and d_v subtrees: $\mathcal{G}_T^{v\to w}$ for $w \in \gamma^v$. Since each subtree has depth at most $\lambda_o - 1$, we can compute $H(\mathcal{G}_T^{v\to w}, w)$ for each subtree $\mathcal{G}_T^{v\to w}$ using equation A.15 (from the inductive hypothesis).

Suppose we impose (any) valid labeling on $\mathcal{G}_T$ starting from v; we refer to the labeled graph as $R_{\mathcal{G},v}$. Given $R_{\mathcal{G},v}$, we order the subtrees of a node in ascending order of their numeric labels. For any fixed ordering of the d_v subtrees of v, we have $\prod_{w\in\gamma^v} H(\mathcal{G}_T^{v\to w}, w)$ nonidentical labelings of $\mathcal{G}_T$ that respect the ordering of subtrees and are isomorphic to any given realization $R_{\mathcal{G}_T,v}$. At most, there can be $d_v!$ arrangements of the subtrees. However, some of the subtrees are isomorphic, so this value over-counts the number of distinct arrangements. That is, switching the order of two nonidentical, isomorphic subtrees is the same as preserving the order and changing both subtrees to the appropriate nonidentical, isomorphic subtree; this is already accounted for in the product $\prod_{w\in\gamma^v} H(G_T^{v\to w}, w)$. $\Delta_j^v!$ of the $d_v!$ permutations of v's subtrees permute the jth unique subtree with isomorphisms of itself. As such, the non-redundant number of different arrangements of the subtrees of node v is $\frac{d_v!}{\Delta_0^v!,\Delta_1^v!\ldots\Delta_{k_v}^v!} = \binom{d_v}{\Delta_0^v,\Delta_1^v,\ldots,\Delta_{k_v}^v}$. This gives the expression in Equation (A.15). □

Step 2. We want to show that

$$\mathbb{P}(C_{\mathcal{G}_T,v}) = \left(\prod_{v\in G_T} \mathbb{P}_D(d_v)\right) \frac{H(\mathcal{G}_T, v_T)|\chi_{\mathcal{G}_T,v}|}{d_{v_T} \prod\limits_{\substack{w\in\phi(v,v_T)\\ \setminus\{v,v_T\}}} (d_w - 1)}.$$

Since $\mathbb{P}(C_{\mathcal{G}_T,v}) = \left(\prod_{v\in G_T} \mathbb{P}_D(d_v)\right) Q(\mathcal{G}_T, v)H(\mathcal{G}_T, v)$, this is equivalent to showing that

$$\begin{aligned}\frac{H(\mathcal{G}_T, v)}{H(\mathcal{G}_T, v_T)} &= \frac{|\chi_{\mathcal{G}_T,v}|}{Q(\mathcal{G}_T, v)d_{v_T} \prod\limits_{\substack{w\in\phi(v,v_T)\\ \setminus\{v,v_T\}}} (d_w - 1)} \\ &= \frac{d_v}{d_{v_T}}|\chi_{\mathcal{G}_T,v}|.\end{aligned}$$

The expressions for $H(\mathcal{G}_T, v_T)$ and $H(\mathcal{G}_T, v)$ differ in that the former starts at the virtual source and counts all subtrees by "trickling down" the tree (i.e., $\Delta_0^w = 0$ for all $w \in G_T$), whereas the latter progresses from an infected leaf v to the virtual source, then recurses over the remaining, unvisited subtrees of v_T. Let P_i denote the ith node in the path from v to v_T, which has length ℓ.

We get

$$H(\mathcal{G}_T, v) = \binom{d_{P_1}}{1, d_{P_1} - 1} \times \binom{d_{P_2} - 1}{1, \Delta_1^{P_2} - 1, \ldots, \Delta_{k_{P_2}}^{P_2}} \prod_{w \in \gamma^{P_2} \setminus \{P_1, P_3\}} H(\mathcal{G}_T^{P_2 \to w}, w) \times \ldots \binom{d_{P_{\ell-1}} - 1}{1, \Delta_1^{P_{\ell-1}} - 1, \ldots, \Delta_{k_{P_{\ell-1}}}^{P_{\ell-1}}} \prod_{\substack{w \in \gamma^{P_{\ell-1}} \\ \setminus \{P_{\ell-2}, P_\ell\}}} H(\mathcal{G}_T^{P_{\ell-1} \to w}, w) \times \binom{d_{P_\ell} - 1}{\Delta_1^{P_\ell} - 1, \ldots, \Delta_{k_{P_\ell}}^{P_\ell}} \prod_{w \in \gamma^{P_\ell} \setminus \{P_{\ell-1}\}} H(\mathcal{G}_T^{P_\ell \to w}, w).$$

where each line corresponds to the terms that result from recursively moving up the path from $v = P_1$ to $v_T = P_\ell$. Similarly, we have

$$H(\mathcal{G}_T, v_T) = \binom{d_{P_1} - 1}{d_{P_1} - 1} \times \binom{d_{P_2} - 1}{\Delta_1^{P_2}, \ldots, \Delta_{k_{P_2}}^{P_2}} \prod_{w \in \gamma^{P_2} \setminus \{P_1, P_3\}} H(\mathcal{G}_T^{P_2 \to w}, w) \times \ldots \binom{d_{P_{\ell-1}} - 1}{\Delta_1^{P_{\ell-1}}, \ldots, \Delta_{k_{P_{\ell-1}}}^{P_{\ell-1}}} \prod_{\substack{w \in \gamma^{P_{\ell-1}} \\ \setminus \{P_{\ell-2}, P_\ell\}}} H(\mathcal{G}_T^{P_{\ell-1} \to w}, w) \times \binom{d_{P_\ell}}{\Delta_1^{P_\ell}, \ldots, \Delta_{k_{P_\ell}}^{P_\ell}} \prod_{w \in \gamma^{P_\ell} \setminus \{P_{\ell-1}\}} H(\mathcal{G}_T^{P_\ell \to w}, w).$$

Here we have expanded the expression in terms of the path from v to v_T to make simplification clearer, where v is the node over which we previously computed $H(\mathcal{G}_T, v)$. Computing the ratio of $H(\mathcal{G}_T, v)$ to $H(\mathcal{G}_T, v_T)$, all the rightmost products of each line cancel. We are left with the ratio of the combinatorial expressions, which simplify to

$$\begin{aligned} \frac{H(\mathcal{G}_T, v)}{H(\mathcal{G}_T, v_T)} &= \frac{d_{P_1}}{d_{P_\ell}} \Delta_1^{P_2} \ldots \Delta_1^{P_{\ell-1}} \Delta_1^{P_\ell} \\ &= \frac{d_v}{d_{v_T}} \Delta_1^{v+1} \ldots \Delta_1^{v_T - 1} \Delta_1^{v_T}. \end{aligned} \tag{A.16}$$

Each Δ_1 denotes the number of child subtrees that are identical to the one containing v, for a given root. As such, the product of Δs above is precisely the number of candidates in the class

being considered, or $|\chi_{\mathcal{G}_T,v}|$. That is, since they are indistinguishable in the unlabelled graph, they generate the same family $C_{\mathcal{G}_T,v}$.

Step 3. We have

$$\begin{aligned}
\sum_{v\in\mathcal{C}_{\mathcal{G}_T}} \mathbb{P}(C_{\mathcal{G}_T,v}) = & \sum_{v\in\mathcal{C}_{\mathcal{G}_T}} \left(\prod_{w\in G_T} \mathbb{P}_D(d_w)\right) H(\mathcal{G}_T, v_T) \\
& \times \frac{|\chi_{\mathcal{G}_T,v}|}{d_{v_T}\prod_{w\in\phi(v,v_T)\setminus\{v,v_T\}}(d_w-1)} \\
= & \left(\prod_{w\in G_T} \mathbb{P}_D(d_w)\right) H(\mathcal{G}_T, v_T)\times \\
& \sum_{v\in\mathcal{C}_{\mathcal{G}_T}} \frac{|\chi_{\mathcal{G}_T,v}|}{d_{v_T}\prod_{w\in\phi(v,v_T)\setminus\{v,v_T\}}(d_w-1)} \\
= & \left(\prod_{w\in G_T} \mathbb{P}_D(d_w)\right) H(\mathcal{G}_T, v_T)\times \\
& \sum_{v\in\partial G_T} \frac{1}{d_{v_T}\prod_{w\in\phi(v,v_T)\setminus\{v,v_T\}}(d_w-1)} \qquad \text{(A.17)}
\end{aligned}$$

where (A.17) follows because every leaf in the graph is a candidate source in exactly one class. We wish to show this last summation sums to 1. Consider a random process over G_T. The process starts at the virtual source v_T, and in each timestep it moves one hop away from v_T. It chooses among the (unvisited) children of a node uniformly at random. At time T, the process is necessarily at one of the leaves of G_T, and the probability of landing at a particular leaf v is precisely $\frac{1}{d_{v_T}\prod_{w\in\phi(v,v_T)\setminus\{v,v_T\}}(d_w-1)}$. Therefore, the sum of this quantity over all leaves $v\in\partial G_T$ is 1.

Step 4. Combining the results from steps 3 and 4, we get that

$$\begin{aligned}
& \frac{\mathbb{P}(C_{\mathcal{G}_T,v})/|\chi_{\mathcal{G}_T,v}|}{\sum_{v\in\mathcal{C}_{\mathcal{G}_T}}\mathbb{P}(C_{\mathcal{G}_T,v})} = \\
& \frac{\left(\prod_{w\in G_T}\mathbb{P}_D(d_w)\right)H(\mathcal{G}_T,v_T)}{\left(\prod_{w\in G_T}\mathbb{P}_D(d_w)\right)H(\mathcal{G}_T,v_T)} \times \frac{|\chi_{\mathcal{G}_T,v}|/|\chi_{\mathcal{G}_T,v}|}{d_{v_T}\prod\limits_{\substack{w\in\phi(v,v_T)\\ \setminus\{v,v_T\}}}(d_w-1)} \\
& \frac{1}{d_{v_T}\prod\limits_{w\in\phi(v,v_T)\setminus\{v,v_T\}}(d_w-1)}.
\end{aligned}$$

Proof of Theorem 2.2.3

To facilitate the analysis, we consider an alternative random process that generates unlabeled graphs G'_T according to the same distribution as G_T (i.e., the infected, unlabeled subgraph em-

bedded in $U(G_D^{(T)})$ from the proof of Theorem 2.2.2). For a given degree distribution D and a stopping time T, the new process is defined as a Galton-Watson process in which the set of offsprings at the first time step is drawn from D and the offsprings at subsequent time steps are drawn from $D-1$. At time $t=0$, a given root node v_T draws its degree d_{v_T} from D, and generates d_{v_T} child nodes. The resulting tree now has depth 1. In each subsequent time step, the process traverses each leaf v of the tree, draws its degree from D, and generates $d_v - 1$ children. The random process continues until the tree has depth $T/2$, since under adaptive diffusion, the infected subgraph at even time T has depth $T/2$. Because the probability of detection in Equation (2.17) does not depend on the degrees of the leaves of G_T, the random process stops at depth $T/2$ rather than $T/2+1$. We call the output of this random process G'_T. The distribution of G'_T is identical to the distribution as the previous random process imposed on G_T, which follows from Equation (A.14) in the proof of Theorem 2.2.2. We therefore use G_T to denote the resulting output in the remainder of this proof.

Distribution D is a multinomial distribution with support $\boldsymbol{f} = (f_1, \ldots, f_\eta)$ and probabilities $\boldsymbol{p} = (p_1, \ldots, p_\eta)$. Without loss of generality, we assume $2 \leq f_1 < \ldots < f_\eta$. Let μ_D denote the mean number of children generated by D:

$$\mu_D = \sum_{i=1}^{\eta} p_i(f_i - 1).$$

There are two separate classes of distributions, which we deal with as separate cases.

Case 1: When $p_1(f_1 - 1) > 1$, we claim that with high probability, there exists a leaf node v in ∂G_T such that on the unique path from the root v_T to this leaf v, all nodes in this path have the minimum degree f_1, except for a vanishing fraction. To prove this claim, consider a different graph H_T derived from G_T by pruning large degree nodes:

1. For a fixed, positive c, find t_0 such that $T/2 = t_0 + c\log(t_0)$.
2. Initialize H_T to be identical to G_T.
3. For each node $v \in H_T$, if the hop distance $\delta_H(v, v_T) \leq c\log(t_0)$, do not modify that node.
4. For each node $v \in H_T$, if the hop distance $\delta_H(v, v_T) > c\log(t_0)$ and $d_v > f_1$, prune out all the children of v, as well as all their descendants (Figure A.5).

We claim that this pruned process survives with high probability. The branching process that generates H_T is equivalent to a Galton-Watson process that uses distribution $D-1$ for the first $c\log(t_0)$ generations, and a different degree distribution $D'-1$ for the remaining generations; D' has support $\boldsymbol{f}' = (f_1, 1)$, probability mass $\boldsymbol{p}' = (p_1, 1-p_1)$, and mean number of children $\mu_{D'} = p_1(f_1 - 1)$.

Note that $f_1 \geq 3$ by the assumption that $p_1(f_1 - 1) > 1$. Hence, the inner branching process up to $c\log t_0$ has probability of extinction equal to 0. This means that at a hop distance of t_0

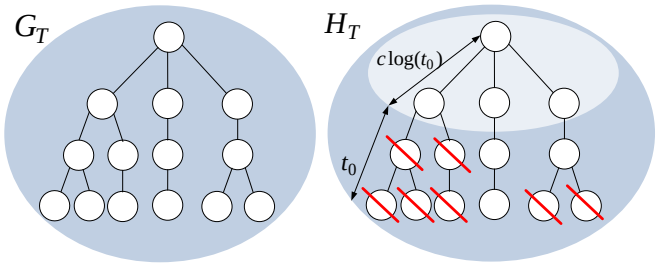


Figure A.5: Pruning of a snapshot. In this example, the distribution D allows nodes to have degree 2 or 3, so we prune all descendants of nodes with degree 3 that are more than $c\log(t_0)$ hops from the root. In this example, $p_1(f_1-1)<1$ and the pruned random process eventually goes extinct.

from v_T, there are at least $(f_1-1)^{c\log(t_0)}$ nodes. Each of these nodes can be thought of as the source of an independent Galton-Watson branching process with degree distribution $D'-1$. By the properties of Galton-Watson branching processes ([57], Thm. 6.1), since $\mu_{D'}>1$ by assumption, each independent branching process' asymptotic probability of extinction is the unique solution of $g_{D'}(s)=s$, for $s\in[0,1)$, where $g_{D'}(s)=p_1\,s^{f_1-1}+(1-p_1)$ denotes the probability generating function of the distribution D'. Call this solution $\theta_{D'}$. The probability of any individual Galton-Watson process going extinct in the first generation is exactly $1-p_1$. It is straightforward to show that $g_{D'}(s)$ is convex, and $g_{D'}(1-p_1)>1-p_1$, which implies that the probability of extinction is nondecreasing over successive generations and upper bounded by $\theta_{D'}$. Then for the branching process that generates H_T, the overall probability of extinction (for a given time T) is at most $\theta_{D'}^{(f_1-1)^{c\log t_0}}$. Increasing the constant c therefore decreases the probability of extinction. If there exists at least one leaf at depth T (i.e., extinction did not occur), then there exists at least one path in H_T of length $t_0-c\log t_0$ in which every node (except possibly the final one) has the minimum degree f_1. This gives

$$\frac{\log(\Lambda_{H_T})}{T/2} \le \frac{t_0\log(f_1-1)+c\log(t_0)\log(f_\eta-1)}{t_0+c\log(t_0)} \tag{A.18}$$

$$\le \log(f_1-1)+\frac{c\log t_0}{t_0}\log\frac{f_\eta-1}{f_1-1}\,, \tag{A.19}$$

with probability at least $1-\theta_{D'}^{(f_1-1)^{c\log t_0}}=1-\theta_{D'}^{t_0^{c\log(f_1-1)}}=1-e^{-C_{D'}t_0}$, where $C_{D'}=\log(\theta_{D'})$ and the upper bound in (A.18) comes from assuming all the interior nodes have maximum degree f_η. Since H_T is a subgraph of a valid snapshot G_T, there exists a path in G_T from the virtual source v_T to a leaf of the tree where the hop distance of the path is exactly $T/2$, and at least t_0 nodes have the minimum degree f_1. Since the second term in (A.19) is $o(t_0)$, the claim follows. The lower bound $\log(\Lambda_{H_T})/(T/2)\ge\log(f_1-1)$ holds by definition. Therefore, for any $\delta>0$, by setting T (and consequently, t_0) large enough, we can make the second term in (A.19) arbitrarily small. Thus, for $T\ge C'_{D,\delta}$, where $C'_{D,\delta}$ is a constant that depends only on the degree distribution and δ, the result holds.

Case 2: Consider the case when $p_1(f_1 - 1) \leq 1$. By the properties of Galton-Watson branching processes ([57], Thm. 6.1), the previous pruned random process that generated graphs H_T goes extinct with probability approaching 1. This implies that with high probability there is no path from the root to a leaf that consists of only minimum degree nodes.

Instead, we introduce a Galton-Watson process with multiple types, derived from the original process. Our approach is to assign a numeric type to each node in G_T according to the number of non-minimum-degree nodes in the unique path between that node and the virtual source. If a node's path to v_T contains too many nodes of high degree, then we prune the node's descendants. The challenge is to choose the smallest pruning threshold that still ensures the pruned tree will survive with high probability. Knowing this threshold allows us to precisely characterize Λ_{G_T} for most of the instances.

To simplify the discussion, we start by considering a special case in which D allows nodes to take only two values of degrees, i.e., $\eta = 2$. We subsequently extend the results for $\eta = 2$ to larger, finite values of η. With a slight abuse of a notation, consider a new random process H_T derived from G_T by pruning large degree nodes in the following way:

1. For a fixed, positive c, find t_0 such that $T/2 = t_0 + c\log(t_0)$.
2. Initialize H_T to be identical to G_T.
3. For each node $v \in H_T$, if the hop distance $\delta_H(v, v_T) \leq c\log(t_0)$, do not modify that node, and assign it type 0.
4. For each node $v \in H_T$, if the hop distance $\delta_H(v, v_T) > c\log(t_0)$, assign v a type ξ_v, which is the number of nodes in $\phi(w, v) \setminus \{v\}$ that have the maximum possible degree f_2, where w is the closest node in H_T to v such that $\delta_H(w, v_T) \leq c\log(t_0)$ (Figure A.6).
5. Given a threshold $r \in (0, 1)$, if a node v has type $\xi_v \geq rt_0$, prune out all the descendants of v. For example, in Figure A.6, if $t_0 = 2$ and the threshold is $r = 0.5$, we would prune out all descendants of nodes with $\xi_v \geq 1$.

We show that for an appropriately-chosen threshold r, this pruned tree survives with high probability. By choosing the smallest possible r, we ensure that Λ_{H_T} consists (in all but a vanishing fraction of nodes) of a fraction r nodes with maximum degree, and $(1 - r)$ of minimum degree. This allows us to derive the bounds on $\log(\Lambda_{H_T})/(T/2)$ stated in the claim, which hold with high probability.

Let $k \equiv rt_0$. The process that generates H_T is equivalent to a different random branching process that generates nodes in the following manner: set the root's type $\xi_{v_T} = 0$. At time $t = 0$, the root v_T draws a number of children according to distribution D, and generates d_{v_T} children, all type 0. Each leaf generates type 0 children according to child degree distribution $D - 1$ until $c\log(t_0)$ generations have passed. At that point, each leaf v in this branching process (which necessarily has type 0) reproduces as follows: if its type $\xi_v > k$, then v does not reproduce. Otherwise, it either generates $(f_1 - 1)$ children with probability p_1, each with state ξ_v, or it generates $(f_2 - 1)$

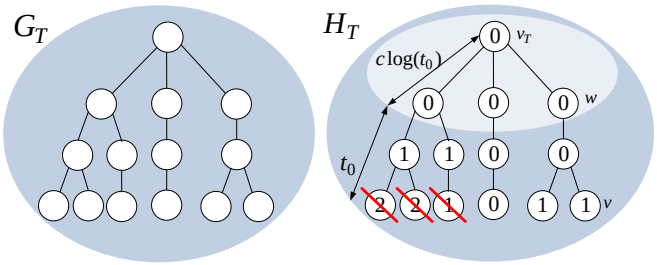


Figure A.6: Pruning of a snapshot using multiple types. In this example, the distribution D allows nodes to have degree 2 or 3. We take $t_0 = 2$ and $r = 0.5$, so all descendants of nodes with type $rt_0 = 1$ are pruned.

children with probability p_2, each with state $\xi_v + 1$. This continues for t_0 generations. Mimicking the notation from Case 1, we use D' to denote the distribution that gives rise to this modified, multi-type random process (in the final t_0 generations); this is a slight abuse of notation since the branching dynamics are multi-type, not defined by realizations of i.i.d. degree random variables.

Lemma 6. *Consider a Galton-Watson branching process with child degree distribution $D - 1$, where each node has at least one child with probability 1, and $\mu_{D-1} > 1$. Then the number of leaves in generation t, $Z^{(t)}$, satisfies the following:*

$$Z^{(t)} \geq e^{C_\ell t}$$

with probability at least $1 - e^{C'_\ell t}$, where both C_ℓ and C'_ℓ are constants that depend on the degree distribution.

(Proof in Section A)

The first $c\log(t_0)$ generations ensure that with high probability, we have at least $e^{C_\ell \log t_0}$ independent multi-type Galton-Watson processes originating from the leaves of the inner subgraph; this follows from Lemma 6. Here we have encapsulated the constant c from the first $c\log(t_0)$ generations in the constant C_ℓ. For example, in Figure A.6, there are 3 independent Galton-Watson processes starting at the leaves of the inner subgraph. We wish to choose r such that the expected number of new leaves generated by each of these processes, at each time step, is large enough to ensure that extinction occurs with probability less than one. For brevity, let $\alpha \equiv p_1(f_1 - 1)$ and let $\beta \equiv p_2(f_2 - 1)$. Let $x^{(t)}$ denote the $(k+1)$-dimensional vector of the expected number of leaves generated with each type from 0 to k in generation t. This vector evolves according to the

following $(k+1) \times (k+1)$ transition matrix M:

$$x^{(t+1)} = x^{(t)} \underbrace{\begin{bmatrix} \alpha & \beta & & & \\ & \ddots & & & \\ & & \ddots & \alpha & \beta \\ & & & & 0 \end{bmatrix}}_{M}.$$

The last row of M is 0 because a node with type k does not reproduce. Since the root of each process always has type 0, we have $x^{(0)} = \boldsymbol{e}_1$, where $\boldsymbol{e}_1$ is the indicator vector with a 1 at index 1 and zeros elsewhere.

Let $Z^{(t)}$ denote the expected number of new leaves created in generation t. This gives

$$\mathbb{E}[Z^{(t)}] = \boldsymbol{e}_1 M^t \mathbb{1}_{(k+1)}^\intercal, \tag{A.20}$$

where $^\intercal$ denotes a transpose, and $\mathbb{1}_{(k+1)}$ is the $(k+1)$ all-ones vector. When $t < k$, this is a simple binomial expansion of $(\alpha+\beta)^t$. For $t \geq k$, this is a truncated expansion up to k:

$$\mathbb{E}[Z^{(t)}] = \sum_{i=0}^{k} \binom{t}{i} \alpha^{t-i} \beta^i. \tag{A.21}$$

We seek the necessary and sufficient condition on r for non-extinction, such that $(1/t)\log(\mathbb{E}[Z^{(t)}]) > 0$. Consider a binomial random variable W with parameter $\beta/(\alpha+\beta) = \beta/\mu_D$ and t trials. Equation (A.21) implies that for large t,

$$\begin{aligned} \mathbb{E}[Z^{(t)}] &= (\alpha+\beta)^t \, \mathbb{P}(W \leq k). & \text{(A.22)} \\ &= \mu_D^t \, \exp\Big\{ -t \, D_{KL}\Big(r \,\|\, \frac{\beta}{\mu_D}\Big) + o(t) \Big\}, & \text{(A.23)} \end{aligned}$$

by Sanov's theorem [29]. Here $D_{KL}(r\|\beta/\mu_D)$ denotes the Kullback-Leibler divergence between the two Bernoulli distributions defined by r and β/μ_D, such that $D_{KL}(r\|\beta/\mu_D) = (1-r)\log((1-r)/(\alpha/\mu_D)) + r\log(r/(\beta/\mu_D))$. We wish to identify the smallest r for which $(1/t)\log(\mathbb{E}[Z^{(t)}])$ is bounded away from zero. Such an r is a sufficient (and necessary) condition for the multi-type Galton-Watson process to have a probability of extinction less than 1. To achieve this, we define the following set of r such that Eq. (A.23) is strictly positive, for some $\epsilon > 0$:

$$\mathcal{R}_{\alpha,\beta}(\epsilon) = \big\{ r \mid \log(\mu_D) \geq D_{KL}(r\|\beta/\mu_D) + \epsilon \big\}, \tag{A.24}$$

Suppose we now choose a threshold $r \in \mathcal{R}_{\alpha,\beta}(\epsilon)$. This is the regime where the modified Galton-Watson process with threshold r has a chance for survival. In other words, the probability of extinction $\theta_{D'}$ is strictly less than one. Precisely, $\theta_{D'}$ is the unique solution to $s = g_{D'}(s)$, where $g_{D'}(s)$ denotes the probability generating function of the described multi-type Galton-Watson process. Using the same argument as in Case 1, we can construct a process where the probability of

extinction is asymptotically zero. Precisely, we modify the pruning process such that we do not prune any leaves in the first $c\log(t_0)$ generations. This ensures that with high probability, there are at least $e^{C_\ell \log(t_0)}$ independent multi-type Galton-Watson processes evolving concurrently after time $c\log(t_0)$, each with probability of extinction $\theta_{D'}$. Hence with probability at least $1 - e^{-2C_{D'}t_0}$ (for an appropriate choice of a constant $C_{D'}$ that only depends on the degree distribution D' and the choice of r), the overall process does not go extinct.

Our goal is to find the choice of r with minimum product of degrees $\log(\Lambda_{G_T})/(T/2)$ that survives. We define r_1 as follows:

$$r_1 \equiv \underset{r\in\mathcal{R}_{\alpha,\beta}(\epsilon)}{\arg\min} \quad (1-r)\log(1-f_1) + r\log(1-f_2).$$

Since $\mathcal{R}_{\alpha,\beta}(\epsilon)$ is just an interval and we are minimizing a linear function with a positive slope, the optimal solution is $r_1 = \inf_{r\in\mathcal{R}_{\alpha,\beta}(\epsilon)} r$. This is a choice that survives with high probability and has the minimum product of degrees. Precisely, with probability at least $1 - e^{-C_{D'}T}$, where $C_{D'}$ depends on D' and ϵ, we have that

$$\frac{\log(\Lambda_{G_T})}{T/2} \leq \langle [1-r_1, r_1], \log(\boldsymbol{f}-1)\rangle + \frac{c\log(t_0)}{t_0}\log(f_2-1)$$

where we define the standard inner product $\langle [1-r_1, r_1], \log(\boldsymbol{f}-1)\rangle \triangleq (1-r_1)\log(f_1-1) + r_1\log(f_2-1)$. It follows that

$$\begin{aligned}&\frac{\log(\Lambda_{G_T})}{T/2} - \langle [1-r^*, r^*], \log(\boldsymbol{f}-1)\rangle \leq \\ &(r_1 - r^*)\log\left(\frac{f_2-1}{f_1-1}\right) + \frac{c\log(t_0)}{t_0}\log(f_2-1)\end{aligned} \tag{A.25}$$

By setting ϵ small enough and t_0 large enough, we can make this as small as we want. For any given $\delta > 0$, there exists a positive $\epsilon > 0$ such that the first term is bounded by $\delta/2$. Further, recall that $T/2 = c\log(t_0) + t_0$. For any choice of ϵ, there exists a $t_{D',\epsilon}$ such that for all $T \geq t_{D',\epsilon}$ the vanishing term in Eq. (A.23) is smaller than ϵ. For any given $\delta > 0$, there exists a positive $t_{D',\delta}$ such that $T \geq t_{D',\delta}$ implies that the second term is upper bounded by $\delta/2$. Putting everything together (and setting ϵ small enough for the target δ), we get that

$$\mathbb{P}\Big(\frac{\log(\Lambda_{G_T})}{T/2} \geq \langle [1-r^*, r^*], \log(\boldsymbol{f}-1)\rangle + \delta\Big) \;\leq\; e^{-C_{D',\delta}T} \tag{A.26}$$

for all $T \geq C'_{D',\delta}$, where $C_{D',\delta}$ and $C'_{D',\delta}$ are positive constants that only depend on the degree distribution D' and the choice of $\delta > 0$.

For the lower bound, we define the following set of r such that Eq. (A.23) is strictly negative:

$$\overline{\mathcal{R}}_{\alpha,\beta}(\epsilon) \;=\; \big\{r \mid \log(\mu_D) \leq D_{KL}(r\|\beta/\mu_D) - \epsilon\big\}. \tag{A.27}$$

Choosing $r \in \overline{\mathcal{R}}_{\alpha,\beta}(\epsilon)$ causes extinction with probability approaching 1. Explicitly, $\mathbb{P}(Z^{(t)} \neq 0)$ is the probability of non-extinction at time t, and $\mathbb{P}(Z^{(t)} \neq 0) \leq \mathbb{E}[Z^{(t)}]$. By Equation (A.23), we have

$$\mathbb{E}[Z^{(t)}] \;\leq\; e^{t(\log(\mu_D) - D_{KL}(r\|\beta/\mu_D) + o(t))}$$

where $\log(\mu_D) - D_{KL}(r\|\beta/\mu_D) \leq -\epsilon$. The probability of extinction is therefore at least $1 - \mathbb{E}[Z^{(t)}] \geq 1 - e^{-t(\epsilon + o(t))}$. So defining

$$r_2 \equiv \underset{r \in \overline{\mathcal{R}}_{\alpha,\beta}(\epsilon)}{\arg\max} \;\; (1-r)\log(1-f_1) + r\log(1-f_2),$$

we have

$$\frac{\log(\Lambda_{G_T})}{T/2} \geq \langle [1-r_2, r_2], \log(\boldsymbol{f}-1)\rangle + \frac{c\log(t_0)}{t_0}\log(f_1 - 1)$$

with probability at least $1 - e^{-C_{D',2}T}$ where $C_{D',2}$ is again a constant that depends on D' and ϵ. It again follows that

$$\begin{aligned} &\frac{\log(\Lambda_{G_T})}{T/2} - \langle [1-r^*, r^*], \log(\boldsymbol{f}-1)\rangle \geq \\ &(r_2 - r^*)\log\left(\frac{f_2 - 1}{f_1 - 1}\right) + \frac{c\log(t_0)}{t_0}\log\left(f_1 - 1\right) \;, \end{aligned} \tag{A.28}$$

where $r_2 - r^*$ is strictly negative. Again, for any given $\delta > 0$, there exists a positive $\epsilon > 0$ such that the first term is lower bounded by $-\delta/2$, and for any choice of ϵ, there exists a $t_{D',\epsilon}$ such that for all $T \geq t_{D',\epsilon}$ the vanishing term in Eq. (A.23) is smaller than ϵ. Note that this ϵ might be different from the one used to show the upper bound. We ultimately choose the smaller of the two ϵ values. For any given $\delta > 0$, there exists a positive $t_{D',\delta}$ such that $T \geq t_{D',\delta}$ implies that the second term is lower bounded by $-\delta/2$. Putting everything together (and setting ϵ small enough for the target δ), we get that

$$\mathbb{P}\Big(\frac{\log(\Lambda_{G_T})}{T/2} \leq \langle [1-r^*, r^*], \log(\boldsymbol{f}-1)\rangle - \delta\Big) \;\leq\; e^{-C_{D',\delta}T} \tag{A.29}$$

for all $T \geq C'_{D',\delta}$, where $C_{D',\delta}$ and $C'_{D',\delta}$ are positive constants that only depend on the degree distribution D' and the choice of $\delta > 0$. This gives the desired result.

We now address the general case for D with support greater than two. We follow the identical structure of the argument. The first major difference is that node types are no longer scalar, but tuples. Each node v's type ξ_v is the $(\eta-1)$-tuple listing how many nodes in the path $\phi(w, v) \setminus \{v\}$ had each non-minimum degree from f_2 to f_η, where w is the closest node to v such that $\delta_H(w, v_T) \leq c\log(t_0)$. Consequently, the threshold $\boldsymbol{r} = [r_1, \ldots, r_{\eta-1}]$ is no longer a scalar, but a vector-valued, pointwise threshold on each element of ξ_v. We let $\boldsymbol{k} = [k_1 = r_1 t_0, \ldots, k_{\eta-1} = r_{\eta-1}t_0]$ denote the time-dependent threshold, and we say $\boldsymbol{k} < \xi_v$ if $k_i < (\xi_v)_i$ for $1 \leq i \leq \eta - 1$.

The matrix M is no longer second-order, but a tensor. Equation (A.20) still holds, except M is replaced with its tensor representation. For brevity, let $\alpha = p_1(f_1 - 1)$ and $\beta_i = p_{i+1}(f_{i+1} - 1)$. Let $\tilde{\beta} = \sum_{i=1}^{\eta-1} \beta_i$. Hence, Equation (A.21) gets modified as

$$\mathbb{E}[Z^{(t)}] = \sum_{i_1=0}^{k_1} \cdots \sum_{i_{\eta-1}=0}^{k_{\eta-1}} \binom{t}{i_1, \ldots, i_{\eta-1}} \alpha^{t-\tilde{\beta}} \beta_1^{i_1} \ldots \beta_{\eta-1}^{i_{\eta-1}}. \tag{A.30}$$

Now we consider a multinomial variable W with parameters β_i/μ_D for $1 \leq i \leq \eta - 1$ and t trials. Note that α/μ_D is the 'failure' probability (corresponding to a node of degree f_1); such events do not contribute to the category count, so the sum of parameters is strictly less than 1. As before, equation (A.30) can equivalently be written as

$$\begin{aligned} \mathbb{E}[Z^{(t)}] &= \mu_D^t \, \mathbb{P}(W \leq \boldsymbol{k}) \\ &= \mu_D^t \, \exp\left\{ - t \, D_{KL}\Big(\boldsymbol{r} \,\|\, \left(\frac{\boldsymbol{\beta}}{\mu_D}\right)\Big) + o(t)\right\}, \end{aligned} \tag{A.31}$$

where $\boldsymbol{\beta}/\mu_D$ denotes elementwise division. Here $D_{KL}(\boldsymbol{r}\|\boldsymbol{\beta}/\mu_D)$ denotes the Kullback-Leibler divergence between the two generalized Bernoulli distributions defined by $\boldsymbol{r}$ and $\boldsymbol{\beta}/\mu_D$, such that $D_{KL}(\boldsymbol{r}\|\boldsymbol{\beta}/\mu_D) = (1 - \sum r_i)\log((1 - \sum r_i)/(\alpha/\mu_D)) + \sum_i r_i \log(r_i/(\beta_i/\mu_D))$. Once again, we wish to obtain bounds on $\mathbb{P}(W \leq \boldsymbol{k})$. As before, we define the following set of r such that Eq. (A.31) is strictly positive, for some $\epsilon > 0$:

$$\mathcal{R}_{\alpha,\boldsymbol{\beta}}(\epsilon) = \left\{ \boldsymbol{r} \mid \log(\mu_D) \geq D_{KL}(\boldsymbol{r}\| \left(\frac{\boldsymbol{\beta}}{\mu_D}\right)) + \epsilon \right\}, \tag{A.32}$$

We now choose a threshold $\boldsymbol{r} \in \mathcal{R}_{\alpha,\beta}(\epsilon)$. Using the same argument as before, we can construct a process where the probability of extinction is asymptotically zero. We again do not prune any leaves in the first $c\log(t_0)$ generations. This ensures that with high probability, there are at least $e^{C_\ell \log(t_0)}$ independent multi-type Galton-Watson processes evolving concurrently after time $c\log(t_0)$, each with probability of extinction $\theta_{D'}$. Hence with probability at least $1 - e^{-2C_{D'}t_0}$ (for an appropriate choice of a constant $C_{D'}$ that only depends on the degree distribution D' and the choice of $\boldsymbol{r}$), the overall process does not go extinct.

We define $\boldsymbol{r}_1$ analogously to the $\eta = 2$ case:

$$\boldsymbol{r}_1 \equiv \underset{\boldsymbol{r} \in \mathcal{R}_{\alpha,\boldsymbol{\beta}}(\epsilon)}{\arg\min} \quad \langle \boldsymbol{r}, \log(\boldsymbol{f} - 1) \rangle \,,$$

where we now define $\langle \boldsymbol{r}, \log(\boldsymbol{f} - 1)\rangle \equiv (1 - \sum_i r_i)\log(f_1 - 1) + \sum_{j=1}^{\eta-1} r_j \log(f_{j+1} - 1)$. Therefore with probability at least $1 - e^{-C_{D'}T}$, where $C_{D'}$ depends on D' and ϵ, we have that

$$\frac{\log(\Lambda_{G_T})}{T/2} \leq \langle \boldsymbol{r}_1, \log(\boldsymbol{f} - 1)\rangle + \frac{c\log(t_0)}{t_0} \log(f_\eta - 1).$$

It follows that

$$\frac{\log(\Lambda_{G_T})}{T/2} - \langle \boldsymbol{r}^*, \log(\boldsymbol{f}-1)\rangle \leq$$
$$\sum_{j=1}^{\eta-1}((r_1)_j - r_j^*)\log\left(\frac{f_{j+1}-1}{f_1-1}\right) + \frac{c\log(t_0)}{t_0}\log\left(f_\eta - 1\right) \ . \tag{A.33}$$

By setting ϵ small enough and t_0 large enough, we can make this as small as we want. For any given $\delta > 0$, there exists a positive $\epsilon > 0$ such that each term in the summation in (A.33) is bounded by δ/η. Further, recall that $T/2 = c\log(t_0) + t_0$. For any choice of ϵ, there exists a $t_{D',\epsilon}$ such that for all $T \geq t_{D',\epsilon}$ the vanishing term in Eq. (A.23) is smaller than ϵ. For any given $\delta > 0$, there exists a positive $t_{D',\delta}$ such that $T \geq t_{D',\delta}$ implies that the second term of (A.33) is upper bounded by δ/η. Putting everything together (and setting ϵ small enough for the target δ), we get that

$$\mathbb{P}\Big(\frac{\log(\Lambda_{G_T})}{T/2} \geq \langle \boldsymbol{r}^*, \log(\boldsymbol{f}-1)\rangle + \delta\Big) \ \leq \ e^{-C_{D',\delta}T} \tag{A.34}$$

for all $T \geq C'_{D',\delta}$, where $C_{D',\delta}$ and $C'_{D',\delta}$ are positive constants that only depend on the degree distribution D' and the choice of $\delta > 0$.

For the lower bound, we again define a set of r such that Eq. (A.23) is strictly negative:

$$\overline{\mathcal{R}}_{\alpha,\boldsymbol{\beta}}(\epsilon) \ = \ \Big\{\, \boldsymbol{r} \mid \log(\mu_D) \leq D_{KL}(\boldsymbol{r}\| \left(\frac{\boldsymbol{\beta}}{\mu_D}\right)) - \epsilon \,\Big\}\,. \tag{A.35}$$

Choosing $\boldsymbol{r} \in \overline{\mathcal{R}}_{\alpha,\boldsymbol{\beta}}(\epsilon)$ causes extinction with probability approaching 1. Explicitly, $\mathbb{P}(Z^{(t)} \neq 0)$ is the probability of non-extinction at time t, and $\mathbb{P}(Z^{(t)} \neq 0) \leq \mathbb{E}[Z^{(t)}]$. By Equation (A.23), we have

$$\mathbb{E}[Z^{(t)}] \ \leq \ e^{t(\log(\mu_D) - D_{KL}(\boldsymbol{r}\|\boldsymbol{\beta}/\mu_D) + o(t))}$$

where $\log(\mu_D) - D_{KL}(\boldsymbol{r}\|\boldsymbol{\beta}/\mu_D) \leq -\epsilon$. The probability of extinction is therefore at least $1 - \mathbb{E}[Z^{(t)}] \geq 1 - e^{-t(\epsilon+o(t))}$. So defining

$$\boldsymbol{r}_2 \equiv \underset{\boldsymbol{r}\in\overline{\mathcal{R}}_{\alpha,\beta}(\epsilon)}{\arg\max} \ \langle \boldsymbol{r}, \log(\boldsymbol{f}-1)\rangle \ ,$$

we have

$$\frac{\log(\Lambda_{G_T})}{T/2} \geq \langle \boldsymbol{r}_2, \log(\boldsymbol{f}-1)\rangle + \frac{c\log(t_0)}{t_0}\log(f_1 - 1)$$

with probability at least $1 - e^{-C_{D',2}T}$ where $C_{D',2}$ is again a constant that depends on D' and ϵ. It follows that

$$\frac{\log(\Lambda_{G_T})}{T/2} - \langle \boldsymbol{r}^*, \log(\boldsymbol{f}-1)\rangle \geq$$
$$\sum_{j=1}^{\eta-1}((r_2)_j - r_j^*)\log\left(\frac{f_{j+1}-1}{f_1-1}\right) + \frac{c\log(t_0)}{t_0}\log\left(f_\eta - 1\right) \ . \tag{A.36}$$

where $(r_2)_j - r_j^*$ is strictly negative. Again, for any given $\delta > 0$, there exists a positive $\epsilon > 0$ such that each term in the summation in (A.36) is lower bounded by $-\delta/\eta$, and for any choice of ϵ, there exists a $t_{D',\epsilon}$ such that for all $T \geq t_{D',\epsilon}$ the vanishing term in Eq. (A.23) is smaller than ϵ. We again choose the smaller of the two ϵ values from the upper and lower bound. For any given $\delta > 0$, there exists a positive $t_{D',\delta}$ such that $T \geq t_{D',\delta}$ implies that the second term is lower bounded by $-\delta/\eta$. Putting everything together (and setting ϵ small enough for the target δ), we get that

$$\mathbb{P}\Big(\frac{\log(\Lambda_{G_T})}{T/2} \leq \langle \boldsymbol{r}^*, \log(\boldsymbol{f}-1)\rangle - \delta\Big) \;\leq\; e^{-C_{D',\delta}T} \tag{A.37}$$

for all $T \geq C'_{D',\delta}$, where $C_{D',\delta}$ and $C'_{D',\delta}$ are positive constants that only depend on the degree distribution D' and the choice of $\delta > 0$. This gives the desired result.

Proof of Lemma 6

If $f_1 > 2$, then the claim follows directly, because each leaf generates at least 2 children in each generation.

If $f_1 = 2$, then for parameters $\rho > 0$ and $\lambda > 0$, we use the Markov inequality to get

$$\begin{aligned}\mathbb{P}(Z^{(t)} \leq \rho) &\leq \mathbb{E}[e^{-\lambda Z^{(t)}}]e^{\rho\lambda} \\ &= g_{D-1}^{(t)}(e^{-\lambda})e^{\rho\lambda}\,,\end{aligned}$$

where $g_{D-1}(s) = \mathbb{E}[e^{s(D-1)}]$ is the probability generating function of $D-1$, and $g_{D-1}^{(t)}(s)$ is the t-fold composition of this function. The goal is to choose parameters ρ and λ such that this quantity approaches zero exponentially fast. The challenge is understanding how $g_{D-1}^{(t)}(e^{-\lambda})$ behaves for a given choice of λ.

Figure A.7 illustrates $g_{D-1}(s)$. Because each node always has at least one child, the probability of extinction for this branching process is 0. As such, the probability generating function is convex, with $g_{D-1}(0) = 0$ and $g_{D-1}(1) = 1$. This implies that for any starting point $e^{-\lambda}$, the fixed-point iteration method approaches 0. We characterize the rate at which $g_{D-1}^{(t)}(s_0)$ approaches 0 by separately bounding the rate of convergence in three different regions of s (Figure A.7). First, we choose a starting point $s_0 = e^{-\lambda}$. We pick any value $s_1 < 1$, such that the slope is strictly larger than one, i.e. $g'_{D-1}(s_1) > 1$. There may be multiple points that satisfy this property; we can choose any one of them, since it only changes the constant factor in the exponent. Without loss of generality, we assume that $s_0 > s_1$, since otherwise we can start the analysis from the region III. Then region I consists of all $s \in [s_1, s_0]$. To define s_2, we draw a line segment parallel to the diagonal from s_1. The intersection is defined as $(s_2, g_{D-1}(s_2))$. Region II consists of all $s \in [s_2, s_1)$. Finally, we choose a threshold ϵ, below which we say the process has converged. Then region III consists of all $s \in [\epsilon, s_2)$. We wish to identify a time t that guarantees, for a given ϵ and λ, that $g_{D-1}^{(t)}(e^{-\lambda}) \leq \epsilon$.

To begin, we split the time spent in each region into t_1, t_2, and t_3, with $t_1 + t_2 + t_3 = t$. We first characterize t_1. Note that $g_{D-1}(s_0) \leq 1 - g'_{D-1}(s_1)(1 - s_0)$ for s_0 in region I. This holds because

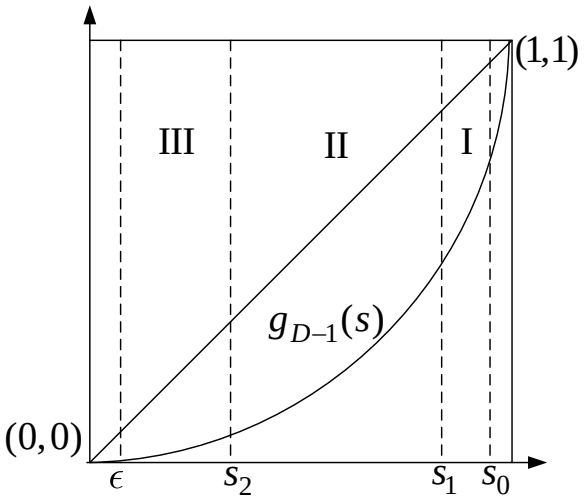


Figure A.7: Regions of the probability generating function, in which we bound the rate of convergence.

s_1 has the lowest slope of all points in region I. Applying this recursively, we get that

$$g_{D-1}^{(t_1)}(s) \;\leq\; \max\left\{1 - g'_{D-1}(s_1)^{t_1}(1-s), g_{D-1}(s_1)\right\}$$

for all s in region I. In region II, we instead upper bound $g_{D-1}(s)$ by the line segment joining $g_{D-1}(s_1)$ and $g_{D-1}(s_2)$. This line has slope 1, giving

$$g_{D-1}^{(t_2)}(s) \;\leq\; \max\left\{g_{D-1}(s_1) - (s_1 - g_{D-1}(s_1))t_2, g_{D-1}(s_2)\right\}.$$

In region III, we upper bound $g_{D-1}(s)$ by the line $y(s) = g'_{D-1}(s_2)s$. We have that $g_{D-1}(s) < g'_{D-1}(s_2) \cdot s$ for s in region III. Recursing this relation gives

$$g_{D-1}^{(t_3)}(s) \;\leq\; \max\left\{g'_{D-1}(s_2)^{t_3} \cdot s, \epsilon\right\}.$$

Thus, if $t \geq 3\max\{t_1, t_2, t_3\}$, then $g_{D-1}^{(t)}(e^{-\lambda}) \leq \epsilon$. In particular, we choose

$$\begin{aligned} t \;\geq\; 3\max\Big\{&\frac{\log((1-g_{D-1}(s_1))/(1-e^{-\lambda}))}{\log(g'_{D-1}(s_1))}, \\ &\frac{g_{D-1}(s_1) - g_{D-1}(s_2)}{s_1 - g_{D-1}(s_1)}, \frac{\log(\epsilon)}{s_2 \log(g'_{D-1}(s_2))}\Big\}. \end{aligned} \tag{A.38}$$

So for sufficiently large t, we have $\mathbb{P}(Z^{(t)} \leq \rho) \leq \epsilon \cdot e^{\rho\lambda}$. By choosing

$$\epsilon = g'_{D-1}(s_2)^{s_2 t/3},$$

we ensure that the third bound on t is always true, and the other two are constant. Similarly, we choose

$$e^{-\lambda} = 1 - \frac{1 - s_2}{g'_{D-1}(s_1)^{t/3}},$$

giving

$$\begin{aligned}
\mathbb{P}(Z^{(t)} \leq \rho) &\leq s_2 \cdot g'_{D-1}(s_2)^{t/3} \left(1 - \underbrace{\frac{1 - s_2}{g'_{D-1}(s_1)^{t/3}}}_{B}\right)^{-\rho} \\
&= s_2 \cdot g'_{D-1}(s_2)^{t/3} (1 - B)^{-\frac{1}{B}B\rho} \\
&\leq s_2 \cdot g'_{D-1}(s_2)^{t/3} e^{\rho(1-s_2)g'_{D-1}(s_1)^{-t/3}}.
\end{aligned}$$

Choosing $\rho = g'_{D-1}(s_1)^{t/3}/(1 - s_2)$, we observe that for t larger than the bound in (A.38), the number of leaves is lower bounded by an exponentially growing quantity (ρ) with probability approaching 1 exponentially fast in t.

Proof of Proposition 2.2.3

Number of nodes. T is either even or odd. At each even T, G_T is a ball (defined over a grid graph) centered at the virtual source with radius $T/2$; that is, G_T consists of all nodes whose distance from the virtual source is at most $T/2$ hops. Thus at each successive even T, G_T increases in radius by one. The perimeter of such a ball (over a two-dimensional grid) is $4\frac{T}{2}$. The total number of nodes is therefore $1 + \sum_{i=1}^{T/2} 4i = \frac{1}{2}(T^2 + 2T + 2)$.

When T is odd, there are two cases. Either the virtual source did not move, in which case $N_T = N_{T+1}$ (because all the spreading occurs in one time step), or the virtual source did move, so spreading occurs over two timesteps. In the latter case, the odd timestep adds a number of nodes that is at least half plus one of the previous timestep's perimeter nodes: $N_T \geq N_{T-1} + 2\frac{T-1}{2} + 1 = \frac{1}{2}(T^2 + 2T + 1)$. This is the smaller of the two expressions, so we have $N_T \geq (T + 1)^2/2$.

Probability of detection. At each even T, G_T is symmetric about the virtual source. We reiterate that the snapshot adversary can only see which nodes are infected—it has no information about who infected whom.

In order to ensure that each node is equally likely to be the source, we want the distribution of the (strictly positive) distance from the virtual source to the true source to match exactly the distribution of nodes at each viable distance from the virtual source:

$$p^{(t)} = \frac{4}{t(\frac{t}{2} + 1)} \begin{bmatrix} 1 \\ 2 \\ \vdots \\ t/2 \end{bmatrix} \in \mathbb{R}^{t/2}. \tag{A.39}$$

There are $4h$ nodes at distance h from the virtual source, and by symmetry all of them are equally likely to have been the source, giving:

$$\begin{aligned}\mathbb{P}(G_T|v^*, \delta_H(v^*, v_t) = h) &= \frac{1}{4h}p_h^{(t)} \\ &= \frac{1}{t(\frac{t}{2}-1)}\,,\end{aligned}$$

which is independent of h. Thus all nodes in the graph are equally likely to have been the source. The claim is that by choosing $\alpha(t,h)$ according to Equation (2.22), we satisfy the distribution in A.39.

The state transition can be represented as the usual $((t/2)+1)\times(t/2)$ dimensional column stochastic matrix:

$$p^{(t+2)} = \begin{bmatrix} \alpha(t,1) & & & \\ 1-\alpha(t,1) & \alpha(t,2) & & \\ & 1-\alpha(t,2) & \ddots & \\ & & \ddots & \alpha(t,t/2) \\ & & & 1-\alpha(t,t/2) \end{bmatrix} p^{(t)}$$

This relation holds because we have imposed the condition that the virtual source never moves closer to the true source. We can solve directly for $\alpha(t,1) = t/(t+4)$, and obtain a recursive expression for $\alpha(t,h)$ when $h > 1$:

$$\alpha(t,h) = \frac{t}{t+4} - \frac{h-1}{h}(1-\alpha(t,h-1))\,. \tag{A.40}$$

We show by induction that this expression evaluates to Equation (2.22). For $h=2$, we have $\alpha(t,2) = \frac{t}{t+4} - \frac{1}{2}\frac{4}{t+4} = \frac{t-2}{t+4}$. Now suppose that Equation (2.22) holds for all $h < h_0$. We then have

$$\begin{aligned}\alpha(t,h_0) &= \frac{t}{t+4} - \frac{h_0-1}{h_0}\left(1 - \frac{t-2(h_0-1)}{t+4}\right) \\ &= \frac{t-2(h_0-1)}{t+4}\,,\end{aligned}$$

which is the claim.

By construction the ML estimator for even T is to choose any node except the virtual source uniformly at random. For odd T, there are two options: either the virtual source stayed fixed or it moved. If the former is true, then spreading occurs in one timestep, so the ML estimator once again chooses a node other than the virtual source uniformly at random. If the virtual source moved, then G_T is symmetric about the edge connecting the old virtual source to the new one. Since the adversary only knows that virtual sources cannot be the true source, the ML estimator chooses one of the remaining $N_T - 2$ nodes uniformly at random. This gives a probability of detection of $1/(N_T-2)$. The claim follows from observing that $N_T \geq \frac{1}{2}(T+1)^2 - 2 = \frac{(T+3)(T-1)}{2}$.

Proof of Proposition 2.5.1

The control packet at spy node s_1 includes the amount of delay at $s_1 = 0$ and all descendants of s_1, which is the set of nodes $\{-1, -2, \ldots\}$. The control packet at spy node s_2 includes the amount of delay at $s_2 = n + 1$ and all descendants of s_2, which is the set of nodes $\{n + 2, n + 3, \ldots\}$. Given this, it is easy to figure out the whole trajectory of the virtual source for time $t \geq T_1$. Since the virtual source follow i.i.d. Bernoulli trials with probability q, one can exactly figure out q from the infinite Bernoulli trials. Also the direction D is trivially revealed.

To lighten the notations, let us suppose that $T_1 \leq T_2$ (or equivalently $T_{s_1} \leq T_{s_2}$). Now using the difference of the observed time stamps $T_{s_2} - T_{s_1}$ and the trajectory of the virtual source between T_{s_1} and T_{s_2}, the adversary can also figure out the time stamp T_1 with respect to the start of the infection. Further, once the adversary figures out T_1 and the location of the virtual source v_{T_1}, the timestamp T_2 does not provide any more information. Hence, the adversary performs ML estimate using T_1, D and q. Let $B(k, n, q) = \binom{n}{k} q^k (1-q)^{n-k}$ denote the pmf of the binomial distribution. Then, the likelihood can be computed for T_1 as

$$\mathbb{P}^{(adaptive)}_{T_1|V^*,Q,D}\big(\, t_1 \big| v^*, q, \ell\big) =$$

$$\begin{cases} q\, B(v^* - \frac{t_1}{2} - 2, \frac{t_1}{2} - 2, q)\, \mathbb{I}_{(v^* \in [2+\frac{t_1}{2}, t_1])} & \text{, if } t_1 \text{ even}\,, \\ B(v^* - \frac{t_1+3}{2}, \frac{t_1-3}{2}, q)\, \mathbb{I}_{(v^* \in [\frac{t_1+3}{2}, t_1])} & \text{, if } t_1 \text{ odd}\,, \end{cases} \tag{A.41}$$

$$\mathbb{P}^{(adaptive)}_{T_1|V^*,Q,D}\big(\, t_1 \big|\, v^*, q, r\big) =$$

$$\begin{cases} 0 & \text{, if } t_1 \text{ even}\,, \\ (1-q)\, B(\frac{t_1-1}{2} - v^*, \frac{t_1-3}{2}, q)\, \mathbb{I}_{(v^* \in [1, \frac{t_1-1}{2}])} & \text{, if } t_1 \text{ odd}\,. \end{cases} \tag{A.42}$$

This follows from the construction of the adaptive diffusion. The protocol follows a binomial distribution with parameter q until $(T_1 - 1)$. At time T_1, one of the following can happen: the virtual source can only be passed (the first equation in (A.41)), it can only stay (the second equation in (A.42)), or both cases are possible (the second equation in (A.41)).

Given T_1, Q and D, which are revealed under the adversarial model we consider, the above formula implies that the posterior distribution of the source also follows a binomial distribution. Hence, the ML estimate is the mode of a binomial distribution with a shift, for example when t_1 is even, ML estimate is the mode of $2 + (t_1/2) + Z$ where $Z \sim \text{Binom}((t_1/2) - 2, q)$. The adversary can compute the ML estimate:

$$\hat{v}_{ML} \;=\; \begin{cases} \frac{T_1+2}{2} + \Big\lfloor q\Big(\frac{T_1-2}{2}\Big)\Big\rfloor & \text{if } T_1 \text{ even, } D = \ell\,, \\ \frac{T_1+3}{2} + \Big\lfloor q\Big(\frac{T_1-1}{2}\Big)\Big\rfloor & \text{if } T_1 \text{ odd, } D = \ell\,, \\ 1 + \Big\lfloor (1-q)\Big(\frac{T_1-1}{2}\Big)\Big\rfloor & \text{if } T_1 \text{ odd, } D = r\,. \end{cases} \tag{A.43}$$

Together with the likelihoods in Eqs. (A.41) and (A.42), this gives

$$\mathbb{P}^{(adaptive)}_{T_1,D|V^*,Q}\big(\, t_1, r, \hat{v}_{ML} = v^* \big| v^*, q\big) =$$

$$\frac{1}{2}(1-q)\, B\Big(\frac{t_1-1}{2} - v^*, \frac{t_1-3}{2}, q\Big)\, \mathbb{I}_{(\hat{v}_{ML} = v^*)}\, \mathbb{I}_{(t_1 \text{ is odd})} \tag{A.44}$$

$$\mathbb{P}^{(adaptive)}_{T_1,D|Q}\big(\, t_1, r, V^* = \hat{v}_{ML} \big| q\big) =$$

$$= \frac{1}{2\,n}(1-q)\,B\Big(\frac{t_1-1}{2} - \hat{v}_{ML}, \frac{t_1-3}{2}, q\Big)\; \mathbb{I}_{(t_1 \text{ is odd})} \tag{A.45}$$

$$\leq \frac{(1-q)}{2\,n}\Big(\frac{\sqrt{2}\,\mathbb{I}_{(t_1 \text{ is odd and } t_1>3)}}{\sqrt{\frac{t_1-3}{2}q(1-q)}} + \mathbb{I}_{(t_1=3)}\Big) \tag{A.46}$$

where $\hat{v}_{ML} = \hat{v}_{ML}(t_1, q, r)$ is provided in (A.43), and the bound on $B(\cdot)$ follows from Gaussian approximation (which gives an upper bound $1/\sqrt{2\pi kq(1-q)}$) and Berry-Esseen theorem (which gives an approximation guarantee of $2 \times 0.4748/\sqrt{kq(1-q)}$) [16], for $k = (t_1-3)/2$. Marginalizing out $T_1 \in \{3, 5, \ldots, 2\lfloor (n-1)/2 \rfloor + 1\}$ and applying an upper bound $\sum_{i=1}^{k} 1/\sqrt{i} \leq 2\sqrt{k+1} - 2 \leq 2\sqrt{k-1} + \sqrt{1/(2(k-1))} - 2 \leq \sqrt{4(k-1)}$, we get

$$\mathbb{P}\big(\, D = r, V^* = \hat{v}_{ML}, T_1 \text{ is odd} \big| Q = q\big) \leq \frac{(1-q)\sqrt{2}}{2\,n\,\sqrt{q(1-q)}}\sqrt{8\,\Big\lfloor \frac{n-1}{2} \Big\rfloor} + \frac{1-q}{2\,n}\,. \tag{A.47}$$

Similarly, we can show that

$$\mathbb{P}\big(\, D = \ell, V^* = \hat{v}_{ML}, T_1 \text{ is odd} \big| Q = q\big) \leq \frac{\sqrt{2}}{2\,n\,\sqrt{q(1-q)}}\sqrt{8\,\Big\lfloor \frac{n-1}{2} \Big\rfloor} + \frac{1}{n}\,, \tag{A.48}$$

$$\mathbb{P}\big(\, V^* = \hat{v}_{ML}, T_1 \text{ is even} \big| Q = q\big) \leq \frac{q\sqrt{2}}{2\,n\,\sqrt{q(1-q)}}\sqrt{8\,\Big\lfloor \frac{n}{2} \Big\rfloor} + \frac{1+q}{2\,n}\,, \tag{A.49}$$

Summing up,

$$\mathbb{P}(V^* = \hat{v}_{ML} | Q = q) \;\leq\; \sqrt{\frac{8}{n\,q\,(1-q)}} + \frac{2}{n}\,. \tag{A.50}$$

Recall Q is uniformly drawn from $[0, 1]$. Taking expectation over Q gives

$$\mathbb{P}(V^* = \hat{v}_{ML}) \;\leq\; \pi\sqrt{\frac{8}{n}} + \frac{2}{n}\,, \tag{A.51}$$

where we used $\int_0^1 1/\sqrt{x(1-x)}dx = \arcsin(1) - \arcsin(-1) = \pi$.

Proof of Theorem 2.3.1

We begin by expanding some points regarding the ML estimator in Algorithm 4 that were omitted in Section 2.3. First, note that it is possible to derive an ML estimate without requiring the presence of a spine spy; the estimator described here uses a spine spy purely for ease of exposition. The omitted details are: (1) given a spy node s, how does the estimator find that spy node's pivot ℓ_s? (2) Why does timing information enable the estimator to disregard any subtree neighboring ℓ_{min} that contains at least one spy?

To answer the first question, consider the first spine spy s_0 and all spies in the feasible subtree. For each spy s in the feasible subtree (none of which lies on the spine), there exists a unique path between s and s_0. There exists a unique node on this path that is both part on the spine and closer to the true source than any other node in the path—this is precisely the pivot node. The estimator uses the observed metadata to infer the pivot, as well as its level in the infected subtree, for each spy in the feasible subtree. This inference proceeds by solving a system of equations:

$$h_{s,\ell_s} + h_{\ell_s,s_0} = |\mathcal{P}(s, s_0)|$$
$$h_{\ell_s,s_0} - h_{s,\ell_s} = T_{s_0} - T_s$$

where $\mathcal{P}(s, s_0)$ denotes the path between s and s_0, $h_{s,\ell_s} = \delta_H(s, \ell_s)$ denotes the distance from spy s_i to the pivot node ℓ_s, and h_{ℓ_s,s_0} is equal to $\delta_H(\ell_s, s_0)$ by construction. This system of equations always has a unique solution; hence the uniqueness of ℓ_s given s and s_0. The first equation holds by construction. The second equation holds because conditioned on the time at which the pivot receives the message T_{ℓ_s}, s_0 receives the message at time $T_{\ell_s} + h_{\ell_s,s_0}$, and s receives it at $T_{\ell_s} + h_{s,\ell_s}$.

Let L denote the set of pivots corresponding to each spy in the feasible subtree; in the example in Figure 2.10, $L = \{1, 2\}$. Define $\ell_{min} = \text{argmin}_{\ell \in L} m_\ell$. That is, ℓ_{min} denotes the pivot closest to the true source in hop distance, i.e., whose level is lowest. Now consider the subtrees of depth $m_{\ell_{min}} - 1$ rooted at the neighbors of ℓ_{min}. The subtree including s_0 cannot contain the true source because we know the message traveled from ℓ_{min} to s_0. The source must therefore lie in one of the remaining $d - 1$ neighbor subtrees, which we refer to as candidate subtrees.

We now argue that the estimator can rule out any candidate subtree of ℓ_{min} that contains at least one spy node. Suppose otherwise: there is a candidate subtree containing a spy s, and the source v^* is contained in that subtree. Then the path $\mathcal{P}(v^*, s)$ cannot pass through ℓ_{min} because ℓ_{min} does not belong to any of its neighboring subtrees by construction. Then there must exist some node ℓ' on the spine such that $|\mathcal{P}(\ell', s)| < |\mathcal{P}(\ell_{min}, s)|$. But this is a contradiction because ℓ_{min} is chosen as the minimum-level pivot across all spies, and each spy has a unique pivot on the spine.

Since we can now rule out candidate subtrees with at least one spy, let $X + 1$, $X \in \mathbb{N}$ be the number of candidate subtrees containing no spies. We use this notation because there will always be at least one candidate subtree with no spies (the one containing the true source). In Figure 2.10, $X = 0$. Thus, the ML estimator chooses one of the leaves in the remaining $X + 1$ candidate subtrees uniformly at random. All remaining nodes in $V \backslash U$ have likelihood 0.

Probability of Detection: We condition on the lowest-level pivot node, ℓ_{min}, giving $\mathbb{P}(\hat{v}_{ML} = v^*) = \sum_{\ell_{min}} \mathbb{P}(\hat{v}_{ML} = v^* | \ell_{min}) \mathbb{P}(\ell_{min})$. Since ℓ_{min} lies on the spine, this is equivalent to conditioning on the distance of ℓ_{min} from the true source.

$$\mathbb{P}(\hat{v}_{ML} = v^*) = \sum_{k=1}^{\infty} \underbrace{\frac{(1-p)^{(|T_{d,k}|-1)}\, p}{|\partial T_{d,k}|}}_{\ell_{min}\ (k^{th}\text{ spine node) is a spy}} + \underbrace{+ (1-p)^{|T_{d,k}|}\mathbb{E}_X\Big[\frac{\mathbb{I}(X \neq d-2)}{(X+1)\,|\partial T_{d,k}|}\Big]}_{\ell_{min}\ (k^{th}\text{ spine node) not a spy}}, \tag{A.52}$$

where $X \sim Binom(d-2, (1-p)^{|T_{d,k}|})$, $|T_{d,k}| = \frac{(d-1)^k-1}{d-2}$ is the number of nodes in each candidate subtree for a pivot at level k, and $|\partial T_{d,k}| = (d-1)^{k-1}$ is the number of leaf nodes in each candidate subtree. Let $w = (1-p)$. We have that

$$\begin{aligned}
&\mathbb{E}_X\Big[\frac{\mathbb{I}(X \neq d-2)}{(X+1)\,|\partial T_{d,k}|}\Big] \\
&= \frac{1}{|\partial T_{d,k}|}\left(\mathbb{E}_X\Big[\frac{1}{X+1}\Big] - \frac{1}{d-1}w^{|T_{d,k}|\cdot(d-2)}\right) \\
&= \frac{\left(\frac{1}{(d-1)w^{|T_{d,k}|}}(1-(1-w^{|T_{d,k}|})^{d-1}) - \frac{1}{d-1}w^{|T_{d,k}|\cdot(d-2)}\right)}{|\partial T_{d,k}|}
\end{aligned}$$

where the last line results from the expression for the expectation of $1/(1+X)$ when X is binomially-distributed. Namely if $X \sim \text{Binom}(\tilde{n}, \tilde{p})$, then $\mathbb{E}[1/(X+1)] = \frac{1}{(\tilde{n}+1)\tilde{p}}(1-(1-\tilde{p})^{\tilde{n}+1})$. Simplifying gives
$P_D =$

$$\begin{aligned}
&= \sum_{k=1}^{\infty} \frac{(d-1)pw^{|T_{d,k}|-1} + 1 - w^{|T_{d,k}|\cdot(d-1)} - (1-w^{|T_{d,k}|})^{d-1}}{(d-1)^k} \\
&= p + \frac{1}{d-2} + \\
&\quad + \sum_{k=1}^{\infty} \frac{pw^{|T_{d,k+1}|-1} - w^{|T_{d,k}|\cdot(d-1)} - (1-w^{|T_{d,k}|})^{d-1}}{(d-1)^k} \\
&= p + \frac{1}{d-2} - \sum_{k=1}^{\infty} \frac{1}{(d-1)^k}\Big[w^{|T_{d,k+1}|} + (1-w^{|T_{d,k}|})^{d-1}\Big].
\end{aligned}$$

where the last line holds because $|T_{d,k+1}| - 1 = |T_{d,k}| \cdot (d-1)$.

Expected hop distance: In the main paper, we lower bounded the expected hop distance by assuming that the estimator guesses the source exactly whenever (a) the pivot ℓ_{min} is a spy node or (b) the estimator chooses the correct candidate subtree. Therefore, if the pivot ℓ_{min} is at level k, we only consider estimates that are exactly $2k$ hops away. The estimator chooses an incorrect candidate subtree with probability $X/(X+1)$.

$$\mathbb{E}[\delta_H(\hat{v}_{ML}, v^*)] \geq$$
$$\sum_{k=1}^{\infty} 2k(1-p)^{|T_{d,k}|}\mathbb{E}_{X_k}\Big[\frac{X_k \cdot \mathbb{I}(X_k \neq d-2)}{(X_k+1)}\Big]. \tag{A.53}$$

If $X_k \sim \text{Binom}(\tilde{n}, \tilde{p})$, where $\tilde{n}$ and $\tilde{p}$ depend on d and k, we have

$$\mathbb{E}_{X_k}\Big[\frac{X_k \cdot \mathbb{I}(X_k \neq \tilde{n})}{(X_k+1)}\Big] =$$
$$\frac{1}{(\tilde{n}+1)\tilde{p}}\left[(1-\tilde{p})^{\tilde{n}} + \tilde{p}(1-(1-\tilde{p})^{\tilde{n}}+\tilde{n}) - 1 - \tilde{n}\tilde{p}^{\tilde{n}+1}\right]$$

Simplifying and substituting $\tilde{p} = (1-p)^{|T_{d,k}|}$ and $\tilde{n} = d-2$ gives the expression in the theorem.

Note that this bound is trivially 0 for $d = 3$, since we ignore all nodes in the correct candidate subtree; when $d = 3$, the source's candidate subtree is the only valid option if ℓ_{min} is not a spy. However, for a fixed p with $d \to \infty$, this lower bound approaches the upper bound of $2(1-p)$.

Obtaining a tighter bound is straightforward, but increases the complexity of the expression. These tighter bounds were used for the plots in the main paper. A tighter bound results from considering the cases when (a) the pivot ℓ_{min} is a spy node or (b) the estimator chooses the correct candidate subtree. In both cases, we ignore all but the most distant estimates. For instance, if ℓ_{min} is on the spine at level k, then the estimate will be at most $2(k-1)$ hops away. Using this rule for both cases (a), we compute the probability of selecting one of the most distant options:

$$a_k \equiv \frac{d-2}{d-1}(1-p)^{|T_{d,k}|(d-1)}$$

and for case (b):

$$b_k \equiv p\frac{d-2}{d-1}(1-p)^{|T_{d,k}|-1}$$

Overall, we get a lower bound of

$$\mathbb{E}[\delta_H(\hat{v}_{ML}, v^*)] \geq \sum_{k=1}^{\infty} 2(kr_k + (k-1)(a_k+b_k))$$

Proof of Proposition 2.3.3

All nodes in $V \backslash U$ have likelihood zero, as discussed in the proof of Theorem 2.3.1. The only randomness in adaptive diffusion spreading occurs when a spine node with uninfected neighbors decides which of its neighbors will be added to the spine next. Thus, the likelihood of a candidate source is the sum of likelihoods of all candidate spines starting at the candidate source. Regardless of which node $u \in U$ is the true source, the spine must pass through ℓ_{min}; since there is a unique path between u and ℓ_{min} over trees, the only feasible spine starting at candidate u must traverse

$\mathcal{P}(u, \ell_{min})$. By the Markov property of the adaptive diffusion spreading mechanism, we only need to consider the likelihood of a candidate spine prior to reaching ℓ_{min}. The propagation of the spine thereafter is conditionally independent of the true source, and therefore equally-likely for all candidates. The maximum likelihood estimator must therefore compute the likelihood of each such candidate sub-spine $\mathcal{P}(u, \ell_{min})$. Since each spine node v chooses one of its uninfected neighbors uniformly at random to be the next spine node, the choice of next spine node is simply $1/(\deg(v) - 1)$. Similarly, the likelihood of candidate source u sending the spine in a particular direction is $1/\deg(u)$. The overall likelihood of a candidate is therefore proportional to the product of these degree terms.

Protocol 8 Grid adaptive diffusion

Input: grid contact network $G = (V, E)$, source v^*, time T
Output: set of infected nodes V_T

$V_0 \leftarrow \{v^*\}$, $h \leftarrow 0$, $v_0 \leftarrow v^*$
$\mathcal{K} \leftarrow \{N, S, E, W\}$ ▷ Cardinality directions
let $k_v(u)$ denote u's direction with respect to v
v^* selects one of its neighbors u at random
$V_1 \leftarrow V_0 \cup \{u\}$, $v_1 \leftarrow u$
$h^H = \mathbb{1}_{\{k_v(u)=E\}} - \mathbb{1}_{\{k_v(u)=W\}}$
$h^V = \mathbb{1}_{\{k_v(u)=N\}} - \mathbb{1}_{\{k_v(u)=S\}}$
let $N^K(u)$ represent u's neighbors in directions $K \subseteq \mathcal{K}$
$V_2 \leftarrow V_1 \cup N^{\mathcal{K}}(u) \setminus \{v^*\}$, $v_2 \leftarrow v_1$
$t \leftarrow 3$
for $t \leq T$ **do**
    v_{t-1} selects a random variable $X \sim U(0, 1)$
    if $X \leq \alpha(t-1, |h^V| + |h^H|)$ **then**
        for all $v \in N(v_{t-1})$ **do**
            Infection Message(G,v_{t-1},v,$\{k_v(v_{t-1})\}$, G_t)
    else
        $K \leftarrow \emptyset$
        if $h^H < 0$ **then**
            $K \leftarrow K \cup \{E\}$
        else if $h^H > 0$ **then**
            $K \leftarrow K \cup \{W\}$
        if $h^V < 0$ **then**
            $K \leftarrow K \cup \{N\}$
        else if $h^V > 0$ **then**
            $K \leftarrow K \cup \{S\}$
        v_{t-1} randomly selects $u \in N^{\mathcal{K}\setminus K}(v_{t-1})$
        $h^H = h^H + \mathbb{1}_{\{k_v(u)=E\}} - \mathbb{1}_{\{k_v(u)=W\}}$
        $h^V = h^V + \mathbb{1}_{\{k_v(u)=N\}} - \mathbb{1}_{\{k_v(u)=S\}}$
        $v_t \leftarrow u$
        for all $v \in N^{\mathcal{K}\setminus\{k_{v_{t-1}}(v)\}}(v_t)\}$ **do**
            Infection Message(G,v_t,v,$\{k_{v_t}(v_{t-1}), k_v(v_t)\}$,$V_t$)
            if $t + 1 > T$ **then**
                break
            Infection Message(G,v_t,v,$\{k_{v_t}(v_{t-1}), k_v(v_t)\}$,$V_t$)
    $t \leftarrow t + 2$
procedure INFECTION MESSAGE(G,u,v,K,V_t)
    if $v \in V_t$ **then**
        for all $w \in N^{\mathcal{K}\setminus K}(v)$ **do**
            Infection Message(G,v,w,K,G_t)
    else
        $V_t \leftarrow V_{t-2} \cup \{v\}$

Appendix B

Proofs and Algorithms from Chapter 3

B.1 Proofs

Lemma 1: Unsynchronized files cause nonzeros in $\hat{\boldsymbol{r}}(0)$

Proof. We know that for index j

$$\begin{bmatrix} r(\alpha_1)_j \\ \dots \\ r(\alpha_d)_j \end{bmatrix} = \begin{bmatrix} H(f_j^{(1)}) & & \\ & \dots & \\ & & H(f_j^{(d)}) \end{bmatrix} \cdot \mathbf{V} \cdot \begin{bmatrix} (a_1)_j \\ \dots \\ (a_d)_j \end{bmatrix} \tag{B.1}$$

where $f_j^{(i)}$ denotes record f_j stored at server i. If f_j is synchronized, $[r(\alpha_1)_j,\ \dots,\ r(\alpha_d)_j]^\intercal = H(f_j)\,[q(\alpha_1)_j,\ \dots,\ q(\alpha_d)_j]^\intercal$ Interpolation gives $\hat{\boldsymbol{r}}(0)_j = H(f_j)\boldsymbol{q}(0) = H(f_j)\cdot 0 = 0$. Now suppose f_j is not synchronized. Let $D = diag([H(f_j^{(1)})\ \dots\ H(f_j^{(d)})])$. Eqn. B.1 gives $\begin{bmatrix} r(\alpha_1)_j \\ \dots \\ r(\alpha_d)_j \end{bmatrix} = \begin{bmatrix} H(f_j^{(1)})q(\alpha_1)_j \\ \dots \\ H(f_j^{(d)})q(\alpha_d)_j \end{bmatrix}$. Interpolating, $H(f_j^{(i)})$ differs across servers, so $\hat{r}(0)_w \neq H(f_j)\boldsymbol{q}(0)_j$ in general. Suppose w.l.o.g. that the dth server is unsynchronized. $\hat{\boldsymbol{r}}(0)_j$ will be nonzero unless $H(f_j^{(d)})$ cancels the other terms, which happens with probability $1/2^{\ell L_h}$, where L_h is the length of the hash in symbols [20]. For collision-resistance, $L_h = c\log_\ell L$; the probability of interpolating a false zero is $\leq 1/2^{\ell c \log_{2^\ell} L}$, and the probability of incorrectly interpolating a zero is $\leq 1/L^c$. □

Lemma 2: Nonzero entries in $\hat{\boldsymbol{r}}(0)$ can be found w.h.p.

Proof. The lemma is true iff a) the number of erroneous coded symbols is $\leq s$ and b) the support of $\hat{\boldsymbol{r}}(0)$ is the indices of the unsynchronized files. Per Algorithm 5, the servers return $\tilde{\boldsymbol{r}}(\alpha_1), \dots, \tilde{\boldsymbol{r}}(\alpha_d)$.

The client obtains vector $\boldsymbol{y}$ by taking

$$\begin{aligned}\boldsymbol{y} &= \left[\tilde{\boldsymbol{r}}(\alpha_1) \quad \dots \quad \tilde{\boldsymbol{r}}(\alpha_d)\right] \cdot (\mathbf{V}^{-1})^\intercal [0 \ \cdots \ 0 \ 1]^\intercal . \\ &= \mathbf{A} \cdot \mathbf{R}^\intercal \cdot (\mathbf{V}^{-1})^\intercal [0 \ \cdots \ 0 \ 1]^\intercal = \mathbf{A} \cdot \hat{\boldsymbol{r}}(0)\end{aligned}$$

If the support of $\hat{\boldsymbol{r}}(0)$ is the unsynchronized indices, $\hat{\boldsymbol{r}}(0)$ is at most s-sparse. $\mathbf{A}$ is a parity-check matrix of a linear, systematic, MDS code. At most s estimated coded symbols are erroneous, and can always be located using the $2s$ parity symbols. If the support of $\hat{\boldsymbol{r}}(0)$ is not the unsynchronized indices, Lemma 1 implies that record(s) are unsynchronized, but the corresponding entries of $\hat{\boldsymbol{r}}(0)$ are still zero, which happens with probability $\leq s_o/L^c$. The probability of recovering all unsynced indices is $\geq 1 - s_o/L^c$. □

Theorem 3: PIR algorithm guarantees (MDS codes)

Proof. Probability of success: From Lemmas 1 and 2, the probability of locating the mis-synchronizations (Phase 1) is $\geq (1 - 1/L^c)^{s_o}$. We need as many zero polynomials as there are unsynchronized records. The number of zero polynomials is binomial with parameter $2^{-\ell_2\kappa}$. Letting $p = 2^{-\ell_2\kappa}$, the probability of identifying all unsynchronized files and having enough zero polynomials to avoid them is $(1 - e^{-2n(p-\gamma(n))^2})(1 - 1/L^c)^{s_o}$, by the Chernoff bound. Allowing the hash length constant c to grow as $c = \log_L n$ when $n \to \infty$, we lower bound this probability by $1 - e^{-2n(p-\gamma(n))^2} - \gamma(n)$. The client can recover the desired records with at least this probability, as Vandermonde submatrices have full rank.

Privacy: Phase 1 is private because it is independent of the desired record. Phase 2 is private iff the likelihood of an arbitrary index i being desired does not depend on the received query vectors. Let $\boldsymbol{q}^{(m)}$ denote the query received by the mth server. We want $\frac{P(w=i|\boldsymbol{q}_i^{(1)}=0,\dots,\boldsymbol{q}_i^{(\kappa)}=0)}{P(w=i|\boldsymbol{q}_i^{(1)},\dots,\boldsymbol{q}^{(\kappa)},\ \exists\ j\ s.t.\ \boldsymbol{q}_i^{(j)}\neq 0)} = 1$. Let M denote the event $\boldsymbol{q}_i^{(1)} = 0, \dots, \boldsymbol{q}_i^{(\kappa)} = 0$ (i.e., all κ servers receive a zero at index i of their query vectors). Let J denote the event $\boldsymbol{q}_i^{(1)}, \dots, \boldsymbol{q}^{(\kappa)}$ such that for at least one j, $\boldsymbol{q}_i^{(j)} \neq 0$. We have $\frac{P(w=i|M)}{P(w=i|J)} = \frac{P(M|w=i)P(J)}{P(J|w=i)P(M)} = 1$ since $P(M|w=i) = P(J|w=i) = 2^{-\ell_2\kappa}$ and $P(M) = P(J)$. □

Corollary 2: Algorithm guarantees using PULSE scheme

Proof. The proof is analogous to that of Lemma 2. By Thm. 1, $\hat{\boldsymbol{r}}(0)$ can be recovered from $\boldsymbol{y}$ with probability $\geq 1 - O(1/m)$. $\hat{\boldsymbol{r}}(0)$ doesn't capture all unsynchronized file indices with probability $\leq \frac{s_o}{L^c}$, so the probability of success is at least $(1 - O(1/m))(1 - \frac{s_o}{L^c})$. □

Observation 1: Privacy lost by zeroing unsynchronized files

Proof. M denotes the event $\boldsymbol{q}_i^{(1)} = 0, \dots, \boldsymbol{q}_i^{(\kappa)} = 0$, and J is some event that is not M. Bayes' rule gives $\frac{P(w=i|M)}{P(w=i|J)} = \frac{P(M|w=i)P(J)}{P(J|w=i)P(M)}$. Since the number of mis-synchronizations is uniformly distributed

between 0 and s, M can occur by chance or because index i was unsynchronized:

$$P(M) = \frac{1}{2^{\ell_2 \kappa}} + \frac{s+1}{2n} - \frac{s+1}{2n2^{\ell_2 \kappa}} = \frac{2n + (2^{\ell_2 \kappa} - 1)(s+1)}{2n2^{\ell_2 \kappa}};$$

since $P(\boldsymbol{q}_i = j, j \neq 0) = \frac{1-P(\boldsymbol{q}_i=0)}{2^{\ell_2 \kappa}-1}$, $P(\boldsymbol{q}_i = j, j \neq 0) = \frac{2n2^{\ell_2 \kappa}-2n-(2^{\ell_2 \kappa}-1)(s+1)}{2n \cdot 2^{\ell_2 \kappa}}$. At most κ servers are colluding, so query entries appear independent. Our expression simplifies to $\frac{1-P(\boldsymbol{q}_i=0)}{(2^{\ell_2 \kappa}-1)P(\boldsymbol{q}_i=0)} = \frac{2n-s-1}{2n+(2^{\ell_2 \kappa}-1)(s+1)}$. For $s = o(n)$, this converges to 1 as $n \to \infty$.[1] □

B.2 Related Algorithms

Protocol 9 A κ-collusion-resistant PIR algorithm [51].

Client:

1: Choose distinct $\alpha_1, \ldots, \alpha_d$ from $GF(2^\ell)$ (e.g., $\alpha_i = i$). Define matrix $\mathbf{V}$, with $V_{ij} = \alpha_{d-i+1}^{\kappa-j+1}$.
2: Uniformly draw vectors $\boldsymbol{a}_0, \ldots, \boldsymbol{a}_{\kappa-1}$, $\boldsymbol{a}_i \in GF(2^\ell)^n$. Define the query as $\boldsymbol{q}(X) = \boldsymbol{a}_0 X^\kappa + \ldots + \boldsymbol{a}_{\kappa-1} X + \boldsymbol{e}_w$.
3: Compute $\boldsymbol{q}(\alpha_i)$, $i \in [d]$, and send to server S_i.

Each honest-but-curious server (S_i, $i \in [d]$)**:**

4: Receive $\boldsymbol{q}(\alpha_i) \in GF(2^\ell)^n$, $i \in [d]$.
5: Return $r(\alpha_i) = \langle \boldsymbol{q}(\alpha_i), \boldsymbol{f} \rangle$ to the client.

Client:

6: Receive $r(\alpha_1), r(\alpha_2), \ldots, r(\alpha_d)$ from the servers; build vector $\boldsymbol{r} = [r(\alpha_1) \;\; r(\alpha_2) \;\; \ldots \;\; r(\alpha_d)]^\intercal$.
7: Compute $f_w = [0 \;\; 0 \;\; \ldots \;\; 1]\mathbf{V}^{-1} \cdot \boldsymbol{r}$.

[1] We assume $\ell_2 > 0$ and $s > 0$.

Protocol 10 PULSE decoding algorithm [84]

for constant number of iterations **do**
  for $i = 1, 4, 7, \ldots, m-1$ **do**
    if $||\boldsymbol{y}_i^{i+2}||_2^2 == 0$ **then**
      bin i is a zeroton
    else
      (singleton, v_p, p)= **SingletonEstimator**($\boldsymbol{y}_i^{i+2}$)
      if (singleton == **True**) **then**
        $\boldsymbol{y} = \boldsymbol{y} - v_p \cdot \mathbf{A}[:, p]$
        Set $\hat{\boldsymbol{r}}(0)_p = v_p$
      else
        bin i is a multiton
return $\hat{\boldsymbol{r}}(0)$
SingletonEstimator($\boldsymbol{y}$)
  singleton = **False**
  Ratio test: $p = \log_\alpha y_2 y_1^{-1}$
  if ($p == \log_\alpha y_3 y_2^{-1} \ \wedge \ p \in [n]$) **then**
    return (**True**, y_1, p)
  return (**False**, 0, 0)

Appendix C

Proofs and Algorithms from Chapter 4

C.1 Proofs

[Proposition 4]

Proof. We must show two things: under the described conditions, *(1)* if document f_i contains fewer than m desired keywords (i.e., $|K_i| < m$), then $\hat{r}_i = 0$, and *(2)* if document f_i contains all m desired keywords (i.e., $|K_i| = m$), then $\hat{r}_i \neq 0$.

First, notice that without loss of generality, we can henceforth assume that the noise bits $a_i = 0$ for $i \in X$. This is because if $a_i = 1$ for any number of $i \in X$, the effect is simply to permute the mapping that determines which servers processes which subset of X.

Part 1: ($|K_i| < m$) Suppose document f_i contains q of the requested keywords, with $q < m$. We wish to determine $\hat{r}_i = \bigoplus_{t\in[d]} r_i^{(t)}$, and show that it equals zero.

Let us first consider the noise-free case. By construction, the ith server counts how many elements of $\mathcal{P}(X)_i$ are contained in the jth document (where $\mathcal{P}(X)_i$ is the ith element of the power set of X, the set of queried keywords). We can express this bitwise sum as follows:

$$\hat{r}_i = \bigoplus_{X'\in\mathcal{P}([q])} \bigoplus_{j=1}^{2^{m-q}} |X'|. \tag{C.1}$$

In the outer sum, we are cycling over each possible subset of the q desired keywords contained in f_i. For each subset X', in the inner summation, we are counting the number of elements of $\mathcal{P}(X)$ that contain exactly X' and not $X \setminus X'$. Since each inner summation fixes exactly q out of m elements, the number of such sets is 2^{m-q}. Recall that for $x \in GF(2^a)$, $x \oplus x = 0$. Therefore, as long as each inner summation has an even number of terms, the entire summation cancels to zero. But this is always the case, because each inner summation has 2^{m-q} terms, and $q < m$.

Now let us consider the noisy case. The only difference is that the summation in (C.1) is

modified to

$$\hat{r}_i = \bigoplus_{X' \in \mathcal{P}([q])} \bigoplus_{j=1}^{2^{m-q}} |X'| + z. \tag{C.2}$$

where z is an arbitrary element of $\mathbb{Z}_{2^a}$ that represents the noise in the servers' responses resulting from $\boldsymbol{a}$. By construction, this noise (a scalar offset) is the same for every server. The same argument as before holds, so this sum necessarily evaluates to zero. This implies that if the ith document does not contain all of the desired keywords in X, then $\hat{r}_j = 0$. Notice that this argument holds for any $m > 0$, which implies that for *any* m, if f_i contains fewer than m keywords, then $\hat{r}_i = 0$.

Part 2: ($|K_i| = m$) We want to show that the summation $\hat{r}_j$ does *not* evaluate to zero. We rewrite the corresponding summation from (C.2) as

$$\hat{r}_i = \bigoplus_{\ell=0}^{m} \bigoplus_{j=1}^{\binom{m}{\ell}} \ell + z. \tag{C.3}$$

We can write the summation in this form because for any subset $X' \subseteq X$, all the elements of X' appear in f_i by assumption. If $\binom{m}{\ell}$ is even, then the corresponding inner sum cancels to zero, by the same logic as Part 1. Lucas' theorem implies that $\binom{m}{\ell}$ is even iff in the binary representation of m and ℓ, at least one bit of ℓ is strictly larger than the corresponding bit of m.

Since we have chosen $m = 2^c$ for some integer $c > 0$, the binary representation of m is simply $\boldsymbol{b} = (b_{c+1}, b_c, \ldots, b_1) = (1, 0, \ldots, 0)$. Thus the only values of ℓ for which the condition in Lucas' theorem is *not* satisfied are $\ell = 0$ and $\ell = m$. Thus we can simplify (C.3) to

$$\hat{r}_i = (m + z) \oplus z.$$

Since $m \neq 0$, this is nonzero, which gives the claim.

□

[Lemma 3]

Proof. The first part of the proof proceeds similarly to that of Proposition 4.

Part 1: ($|K_i| < m$) Mirroring Proposition 4, we have that

$$\hat{r}_i = \bigoplus_{X' \in \mathcal{P}([q])} \bigoplus_{j=1}^{2^{m-q}} h_m(|X'| + z). \tag{C.4}$$

where z is an arbitrary element of $\mathbb{Z}_{2^a}$ that represents the noise in the servers' response vectors caused by $\boldsymbol{a}$. Notice that we now apply the post-processing function $h_m(\cdot)$ to the elements of each $\boldsymbol{r}^{(j)}$. Regardless of the function $h_m(\cdot)$, the same argument from Proposition 4 holds: that is, each

distinct value in the inner summation of (C.4) appears an even number of times, so the overall sum is zero.

Part 2: ($|K_i| = m$) We must show that the summation $\hat{r}_i$ does *not* evaluate to zero. Once again, for this case, we consider the equation

$$\hat{r}_i = \bigoplus_{\ell=0}^{m} \bigoplus_{j=1}^{\binom{m}{\ell}} h_m(\ell + z). \tag{C.5}$$

A sufficient (but by no means necessary) condition that ensures $\hat{r}_i \neq 0$ is for the summation in (C.5) to contain an *odd* number of *odd* elements. If this condition is true, then the ones bit (i.e. the least significant bit of each term in the summation) must appear an odd number of times in summation (C.5), so the total sum is nonzero (regardless of the values of the higher bits). As we will show, the choice of $h_m(\cdot)$ in Lemma 3 is designed to enforce this condition. Notice that we could just as easily required a different bit to appear an odd number of times to enforce the nonzero condition.

We can write (C.5) as

$$\hat{r}_i = \bigoplus_{\ell \in \mathcal{B}} h_m(\ell + z). \tag{C.6}$$

where $\mathcal{B}$ is the set of $\ell \in [m]$ such that $\binom{m}{\ell}$ is odd. Notice that the set $\mathcal{B}$ does not depend on the added noise z. One (sufficient, not necessary) way to enforce our desired nonzero condition is to impose the condition that $\hat{r}_i = 1$, regardless of the value of z, and restrict the range of $h_m(\cdot)$ to $\{0, 1\}$. Using the length-2^a vector $\boldsymbol{h}_m$ to denote $h_m(\cdot)$ over $\mathbb{Z}_{2^a}$, we can write this condition as

$$\boldsymbol{g} * \boldsymbol{h}_m = \mathbf{1}_{2^a} = \hat{\boldsymbol{r}}_i \tag{C.7}$$

where $\hat{\boldsymbol{r}}_i$ is a vector in which the jth entry $(\hat{r}_i)_j$ represents $\hat{r}_i$ when the noise $z = j$ and $*$ denotes a circular correlation defined as

$$(\boldsymbol{g} * \boldsymbol{h}_m)_z \equiv \bigoplus_{\ell=0}^{2^a-1} g_\ell \cdot (h_m)_{\ell+z} = \bigoplus_{\ell=0}^{2^a-1} g_{\ell+z} \cdot (h_m)_\ell.$$

Finally, $\boldsymbol{g}$ is an indicator vector of length 2^a, in which

$$g_i = \begin{cases} 1 & \text{if } i \in \mathcal{B} \\ 0 & \text{otherwise.} \end{cases}$$

Condition (C.7) is simply repackaging (C.6): It says that regardless of the added noise z, we always want the output $(\hat{r}_i)_z = 1$. We must show that the choice of $\boldsymbol{h}_m$ in equation (4.1) satisfies this condition. First, we observe the following:

Observation 2. *For all $d \in \overline{\mathcal{B}}$, $d \neq 0$, it holds that $d < 2^{a-1}$. Moreover, for all $v \in [2^a]$, $g_v + g_{v+d} \leq 1$. That is, at most one of g_v and g_{v+d} can equal 1, but not both. This implies that no two bits in $\boldsymbol{g}$ are exactly d indices apart. Throughout the rest of this proof, we use the zero-indexed notation $[x] = \{0, 1, \ldots, 2^a - 1\}$.*

To show this observation, notice that by definition, the $(a-1)$th bit $b_{a-1} = 1$. Thus by the construction of $\overline{\mathcal{B}}$, $d < 2^{a-1}$. To show the second part of the observation, suppose the contrary: there exists some v such that $g_v = g_{v+d} = 1$. By the construction of $\boldsymbol{g}$, this implies that $v, (v+d) \in \mathcal{B}$, which in turn implies that $d \in \mathcal{B}$. However, d has a unique binary representation. In this binary representation, none of the set bits (i.e., bits equaling one) belong to $\mathcal{O}$, by the definition of $\overline{\mathcal{B}}$. Therefore, $d \notin \mathcal{B}$. The observation follows.

Using this observation, we show that for all $z \in [2^a]$, $(\boldsymbol{g} * \boldsymbol{h}_m)_z = 1$. Define

$$(\boldsymbol{g} *_{\mathbb{Z}} \boldsymbol{h}_m)_z \equiv \sum_{\ell=0}^{2^a-1} g_\ell \cdot (h_m)_{\ell+z} = \sum_{\ell=0}^{2^a-1} g_{\ell+z} \cdot (h_m)_\ell.$$

where $*_{\mathbb{Z}}$ denotes summation over the field $\mathbb{Z}$ rather than $GF(2^a)$. Notice that since both $\boldsymbol{g}$ and $\boldsymbol{h}_m$ are nonnegative vectors, then if $(\boldsymbol{g} *_{\mathbb{Z}} \boldsymbol{h}_m)_z = 1$, it must also hold that $(\boldsymbol{g} * \boldsymbol{h}_m)_z = 1$ (but not necessarily vice versa). We therefore show the former by splitting the proof into two parts: (1) show that $(\boldsymbol{g} *_{\mathbb{Z}} \boldsymbol{h}_m)_z \leq 1$, (2) show that $(\boldsymbol{g} *_{\mathbb{Z}} \boldsymbol{h}_m)_z \geq 1$.

Part 1: $[(\boldsymbol{g} *_{\mathbb{Z}} \boldsymbol{h}_m)_z \leq 1]$ Suppose the opposite is true: $\sum_{\ell=0}^{2^a-1} g_\ell \cdot (h_m)_{\ell+z} > 1$. Then there must exist (at least) two indices j and ℓ such that $(h_m)_j = (h_m)_\ell = g_{j+z} = g_{\ell+z} = 1$. Without loss of generality, assume $j < \ell$. This assumption implies that $j, \ell \in \overline{\mathcal{B}}$ and $(\ell - j) \in \mathcal{B}$.

From the definition of $\overline{\mathcal{B}}$, we can equivalently write

$$\begin{aligned} \ell - j &= \sum_{\ell_1 \in B_1 \in \mathcal{P}(\mathcal{Z})} \ell_1 - \sum_{j_1 \in B_1' \in \mathcal{P}(\mathcal{Z})} j_1 \\ &= \sum_{\ell_2 \in B_2 \in \mathcal{P}(\mathcal{Z})} \ell_2 - \sum_{j_2 \in B_2' \in \mathcal{P}(\mathcal{Z})} j_2 \end{aligned}$$

where $B_2 = B_1 \setminus B_1'$ and $B_2' = B_1' \setminus B_1$. Since elements of $\mathcal{B}$ are defined over $\mathbb{Z}_{2^a}$, the summations and subtraction above is executed over $\mathbb{Z}_{2^a}$ as well. Now we can consider three cases:

- $B_2' = \emptyset$: Then $(\ell - j) \in \overline{\mathcal{B}}$, which is impossible since $(\ell - j)$ is also an element of $\mathcal{B}$.
- $B_2 = \emptyset$: Then $\ell - j = -d$ for some $d \in \overline{\mathcal{B}}$. This is impossible because we have assumed that $\ell > j$ and $j, \ell \in [2^{a-1}]$, and $\forall d \in \overline{\mathcal{B}}$, $d < 2^{a-1}$.
- $B_2 \neq \emptyset$ and $B_2' \neq \emptyset$: In this case, we can equivalently write $\ell - j$ as $(2^{\alpha_1} + \ldots + 2^{\alpha_w}) - (2^{\beta_1} + \ldots + 2^{\beta_y})$ where $\alpha_1 \neq \ldots \neq \alpha_w \neq \beta_1 \neq \ldots \neq \beta_y$. Since B_2' is not empty, there is at least one term 2^{β_1} being subtracted. Since these α and β bit positions are all unequal (i.e., the sets are disjoint), it holds that for at least one bit location β_q, $q \in [y]$ and $\beta_q \in \mathcal{Z}$, the β_qth bit of $\ell - j$ is one. This holds because of basic arithmetic and in turn implies that

at least one component of the binary representation of $\ell - j$ is an element of $\overline{\mathcal{B}}$, But this is impossible since $\ell - j$ is assumed to be an element of $\mathcal{B}$, and their intersection is $\{0\}$.

Therefore, it must hold that $(\boldsymbol{g} *_{\mathbb{Z}} \boldsymbol{h}_m)_z \leq 1$.

Part 2: $[(\boldsymbol{g} *_{\mathbb{Z}} \boldsymbol{h}_m)_z \geq 1]$ Suppose that for some z, $(\boldsymbol{g} *_{\mathbb{Z}} \boldsymbol{h}_m)_z = 0$. This is the only value less than one that $(\boldsymbol{g} *_{\mathbb{Z}} \boldsymbol{h}_m)_z$ can take since the output of the correlation is a nonnegative integer. Notice that $\mathcal{B}$ and $\overline{\mathcal{B}}$ are tiling complements of one another, and they generate a translational tiling of $\mathbb{Z}_{2^a}$; this can be checked using the algorithm from [64], for instance. This means that if we generate a set by circularly shifting the elements of $\mathcal{B}$ by each $b \in \overline{\mathcal{B}}$ over the field $\mathbb{Z}_{2^a}$, the resulting set spans the whole space $[2^a]$, and each element has a *unique* representation in terms of an element of $\mathcal{B}$ and a translation of that element by another element $b \in \overline{\mathcal{B}}$. The condition $(\boldsymbol{g} *_{\mathbb{Z}} \boldsymbol{h}_m)_z = 0$ implies that the circularly shifted $\sigma_z(\boldsymbol{h}_m)$ overlaps with *none* of the ones in $\boldsymbol{g}$, i.e., $\nexists \ell$ such that $g_{\ell+z} \cdot (h_m)_\ell = 1$. Then by the pigeonhole principle, there must exist another shifted $\sigma_{z_2}(\boldsymbol{h}_m)$ that is part of a tiling that overlaps with at least *two* ones in $\boldsymbol{g}$. But we saw in Case 1 that this is impossible. So $(\boldsymbol{g} *_{\mathbb{Z}} \boldsymbol{h}_m)_z \geq 1$.

Since $(\boldsymbol{g} *_{\mathbb{Z}} \boldsymbol{h}_m)_z \geq 1$ and $(\boldsymbol{g} *_{\mathbb{Z}} \boldsymbol{h}_m)_z \leq 1$, it must hold that $(\boldsymbol{g} *_{\mathbb{Z}} \boldsymbol{h}_m)_z = 1$ for all z.

□

[Theorem 4.3.1]

Proof. There are two parts to show. The first is correctness with the desired probability and communication cost. The second is the privacy guarantees.

Correctness: Lemma 3 guarantee that the post-processed reply vectors are additively sparse, and their support is exactly equal to the desired documents that feature all keywords. The linearity of the compression scheme implies that the client can perfectly recover the compressed (sparse) vector $\hat{\boldsymbol{r}}$ from the individually compressed (non-sparse) vectors $\tilde{\boldsymbol{r}}^{(i)}$ produced by each server. Then by the properties of the encoding scheme in [84], the signal can be perfectly reconstructed with probability at least $1 - O(s^{-3/2})$ using $O(s)$ communication cost (Theorem 2.3.1 from [84]).

Privacy: The data sent to each server is the sum of a true signal and Bernoulli(1/2) noise. Therefore, the information observed by the server is distributed as Bernoulli(1/2) noise, regardless of the client's query. Thus the mutual information between the desired set X and any observed query $\boldsymbol{q}^{(i)}$ is zero. This gives an information-theoretic privacy guarantee against an honest-but-curious adversary. □

C.2 Algorithms

Protocol 11 A multi-server, conjunctive query private search algorithm for general number of queries m, with rankings. Input: Queried keywords $X = \{k_{i_1}, \ldots, k_{i_m}\}$. Output: List of indices of documents featuring all keywords in X.

Client:

Uniformly draw noise vector $\boldsymbol{a}$, $a_i \sim Bern(\frac{1}{2})$, $i \in [n]$
Generate the power set of X, $\mathcal{P}(X) = \{X_1, \ldots, X_{2^m}\}$
Define the ith query as $\boldsymbol{q}^{(i)} = \boldsymbol{a} + \boldsymbol{e}_{X_i}$
Send $\boldsymbol{q}^{(i)}$, $i \in [d]$, to server S_i

Each honest-but-curious server (S_i, $i \in [d]$)**:**

Initialize $\boldsymbol{r}^{(i)} = \mathbf{0}_n$
for $j \in [|\mathcal{K}|]$ **do**
    if $q_j^{(i)} = 1$ **then**
        for $s \in \mathcal{S}_j$ **do**
            $r_s^{(i)} = (r_s^{(i)} + 1) \mod 2^a$
Let $\tilde{\boldsymbol{r}}^{(i)} = h_m(\boldsymbol{r}^{(i)})$
Return $\boldsymbol{y}^{(i)} = \mathbf{A} \cdot \tilde{\boldsymbol{r}}^{(i)}$, computed over $GF(2^a)$

Client:

Compute $\boldsymbol{y} = \bigoplus_{j=1}^{d} \boldsymbol{y}^{(i)}$
Reconstruct $\hat{\boldsymbol{r}}$ from $\boldsymbol{y}$ and $\mathbf{A}$, e.g. using [84]
Return the nonzero indices of $\hat{\boldsymbol{r}}$