Dartmouth College

Dartmouth Digital Commons

Dartmouth College Ph.D Dissertations

Theses and Dissertations

12-14-2007

Settling for limited privacy: how much does it help?

Anna M. Shubina
Dartmouth College

Settling for limited privacy: how much does it help?

Anna Shubina

Advisor: Amit Chakrabarti

Department of Computer Science

Dartmouth College

Hanover, NH 03755

USA

Dartmouth Computer Science Technical Report

TR2007-609

December 14, 2007

Abstract

This thesis explores practical and theoretical aspects of several privacy-providing technologies, including tools for anonymous web-browsing, verifiable electronic voting schemes, and private information retrieval from databases. State-of-art privacy-providing schemes are frequently impractical for implementational reasons or for sheer information-theoretical reasons due to the amount of information that needs to be transmitted. We have been researching the question of whether relaxing the requirements on such schemes, in particular settling for imperfect but sufficient in real-world situations privacy, as opposed to perfect privacy, may be helpful in producing more practical or more efficient schemes.

This thesis presents three results. The first result is the introduction of caching as a technique for providing anonymous web-browsing at the cost of sacrificing some functionality provided by anonymizing systems that do not use caching. The second result is a coercion-resistant electronic voting scheme with nearly perfect privacy and nearly perfect voter verifiability. The third result consists of some lower bounds and some simple upper bounds on the amount of communication in nearly private information retrieval schemes; our work is the first in-depth exploration of private information schemes with imperfect privacy.

Contents

List of Tables

List of Figures

Acknowledgments

First and foremost, I would like to thank my husband, Sergey Bratus, without whose support this thesis would have never been written. I would also like to thank my parents and my sister for all those things that only family can provide.

I am grateful to my advisor, Amit Chakrabarti, for his guidance, help, and for the introduction to the field. I am also grateful to my former advisor and committee member, Sean Smith, for supervising and guiding me for most of my time at Dartmouth College. I would also like to thank Sean for his continued encouragement to define and pursue new research topics.

I would like to thank my committee members, Peter Winkler and Jay Aslam, for all their help with my thesis. I am especially grateful to Jay for finding the time in his incredibly busy teaching schedule to shoulder the additional burden of being an external committee member.

I would like to extend thanks to the PKI lab and to the Theory Reading Group for useful discussion. I am grateful also to Andrew Campbell for his help and encouragement. Finally, my thanks go to the staff of the Computer Science Department of Dartmouth College - to Kelly Clark and Nancy Wendlandt - for being extremely helpful, supportive, and sympathetic.

This work was supported in part by the NSF CAREER Award (CCF-0448277), by the Mellon Foundation, by the NSF (CCR-0209144), by Internet2/AT&T, and by the Office for Domestic Preparedness, U.S. Dept of Homeland Security (2000-DT-CX-K001). The views and conclusions do not necessarily represent those of the sponsors.

Introduction

This thesis explores practical and theoretical aspects of several privacy-providing technologies, including tools for anonymous web-browsing, verifiable electronical voting schemes, and private information retrieval from databases. State-of-the-art privacy-providing schemes are frequently impractical for implementational or sheer information-theoretical reasons. We have been researching the question of whether relaxing the requirements on such schemes, in particular settling for imperfect but sufficient in certain real-world situations privacy as opposed to perfect privacy, may be helpful in producing more practical or more efficient schemes.

This thesis presents updated and extended versions of three papers: “Using caching for browsing anonymity” by Anna Shubina and Sean Smith [SS03], “Design and prototype of a coercion-resistant, voter verifiable electronic voting system” by Anna Shubina and Sean Smith [SS04], and “Nearly private information retrieval” by Amit Chakrabarti and Anna Shubina [CS07].

In Chapter 1 we introduce caching as a technique for providing anonymous web-browsing as presented in [SS03]. Web caches are quite common but generally not used for anonymization despite having useful properties that other anonymizing systems lack. This chapter also contains an overview of various techniques that solve the same problem and points out their deficiencies. We describe our prototype, which uses Google cache. This work was done under the supervision of Sean Smith.

Chapter 2 covers our work on coercion-resistant verifiable electronic voting as presented in [SS04]. In this chapter we describe an electronic voting scheme that addresses the contradiction between verifiability and receipt-freeness (being unable to prove to anyone how the voter voted) by relying on the information in the voter's head. In this scheme, privacy of the voter's choices is not perfect but likely to be sufficient in a real-world situation. We describe our prototype. This work was also done under the supervision of Sean Smith.

Chapter 3 presents an extended version of our paper [CS07] on nearly private information retrieval. Before our work on nearly private information retrieval, almost all of the bounds for communication complexity on private information schemes assumed that the schemes leak no information about the index requested. We were able to obtain some lower and upper bounds on schemes that allow leaking a small amount of information.

In Chapter 3 the following results in nearly private information retrieval are presented.

- We derive a simple upper bound in the 1-server case. This bound is essentially optimal. We use this bound to obtain an upper bound in the 2-server case.

- We prove a linear lower bound in the 1-server case.

- We prove a linear lower bound in the 2-server case. This bound is an improvement over a lower bound shown in [GKST02] by Goldreich et al.

Chapter 1

Using caching for browsing anonymity

In the first part of my graduate work I was studying tools that provide anonymity on the internet.

Privacy-providing tools, including tools that provide anonymity, are gaining popularity in the modern world. The goals of the users of such tools vary widely, from avoiding tracking and profiling to avoiding internet censorship. The reactions of businesses to the spread of such tools vary widely as well: whereas some businesses are unhappy with the growth of privacy-enhancing technologies, others can use lack of information about their users to avoid unnecessary liability and even possible harassment by parties with contrary business interests and to gain a competitive market edge.

An interesting subproblem of the more general problem of providing anonymity on the internet is the problem of anonymous web-browsing. A system allowing anonymous web-browsing should respond in real time, as opposed to, for example, anonymous e-mail that allows delayed traffic. The requirement of real-time responses makes certain attacks much easier than they would have been otherwise.

Currently, users interested in anonymous web-browsing have to choose between single-hop proxies and the few more complex systems that are available. The single-hop proxies leave the user to rely on the provider of the service for not disclosing his data and also leave him or her vulnerable to adversaries watching both the traffic before the proxy and the traffic at the destination. The more complex systems have more points where they can be attacked and are arguably vulnerable (although, possibly harder to attack) in more scenarios. For example, the state-of-the-art system in the field of anonymous web-browsing is Tor (`tor.eff.org`). Tor works by letting the user obtain the list of routers from the directory server, having the user select a path of 3 random nodes and route his traffic to the destination through the path. In this scenario the adversary can be almost anywhere – on the servers, between them, at the destination, between the user and the first node, perhaps anywhere except the directory servers which are supposed to be honest (although the machines on the path to them may not be). What is even worse, a sophisticated adversary would be able to record traffic day after day for later analysis, and since the real users tend to have distinguishable web-browsing patterns, such long-term recording puts them at risk of deanonymization. In the situation where the adversary is able to watch many nodes, including a number of entrance and exit points, this kind of attack (called *the long-term intersection attack*) becomes a major problem not fully addressed by any existing real-time web-browsing anonymizing systems.

In the paper "Using caching for browsing anonymity" (A.M. Shubina and S.W. Smith, 2003), we suggested to use for anonymization a well-known scheme that was not commonly used in that manner. Our idea took care of long-term intersection attacks by putting some limitations on the user's web-browsing. Namely, we proposed and prototyped a caching proxy system for allowing users to retrieve data from the World-Wide Web in a way similar to many web caches such as Google (`google.com`) or WayBack Machine (`archive.org`). The system works by allowing the user to see only the cached versions of all the data, retrieved by the caching proxy independently of the user activity on the proxy. In this system the actual destination and any observers before the destination see no traffic that can be correlated with the user's activity. If the user's traffic going to the caching proxy is encrypted, an observer between the user and the caching proxy sees nothing except whatever patterns can be glimpsed from the encrypted traffic. Thus our system provides recipient unobservability by a third party and sender unobservability by the recipient, disposing with intersection attacks.

We built a prototype of such a caching proxy by rerouting all user requests to Google cache (which, unfortunately, does not accept encrypted traffic).

1.1 Available anonymizing services

Following the recent political and economic developments, from increased consumer profiling for many ends and purposes (see, for example, [Ele03]) to plans of the US Government for Terrorism Information Awareness (formerly Total Information Awareness, see [DAR03]), a number of previously unconcerned people started to contemplate being more careful with exposing or giving out any information about themselves. A number of businesses that have been collecting data about their customers have also taken damage from adversary parties. It is no longer a purely hypothetic possibility that a vendor of electronic equipment will find his web logs subpoenaed on murky legal grounds. An example is the DirecTV case, where customer lists of a number of different companies that sold smart cards were seized, and people on these lists accused of piracy [Pou03].

The lack of privacy of transactions on the web being well-known (as documented, for example, at [Ele97]), and more people are now starting to try services that claim to provide anonymous browsing on the web, such as `anonymizer.com` or `the-cloak.com`.

There currently exist a number of very different services - free or paid, with more or fewer features and paranoia, with better or worse performance, with bigger or smaller quotas - which claim that they would anonymize the web browsing of their users. Looking closer, however, one cannot help noticing that most of them are based on the same underlying assumptions, the same attacker model, and the same design.

The few more complicated systems that consider a more powerful attacker require extra software download and setup from the user (and some of them may also come with extra obligations, such as automated request-forwarding in case of Crowds [RR98]). Of these more complicated systems, we are aware of only three that are currently publicly available: Tor [DMS04], homepage at `http://tor.eff.org`, JAP [BFK01], available at `http://anon.inf.tu-dresden.de/index_en.html`, and I2P, homepage at `http://www.i2p.net`.

The first commercial mix-net, Freedom Network [GS01] went offline permanently [SG00] in 2001; the homepage of Crowds is currently offline, and the resource itself appears to be unavailable. Although Tor has gained public appreciation and is gaining servers and users, its installation may not be an option for many users who are not fully in control of their computers; it is also far from invulnerable to traffic analysis, as demonstrated, for example, by Murdoch and Danezis [MD05]. JAP and I2P come with the same caveats as Tor.

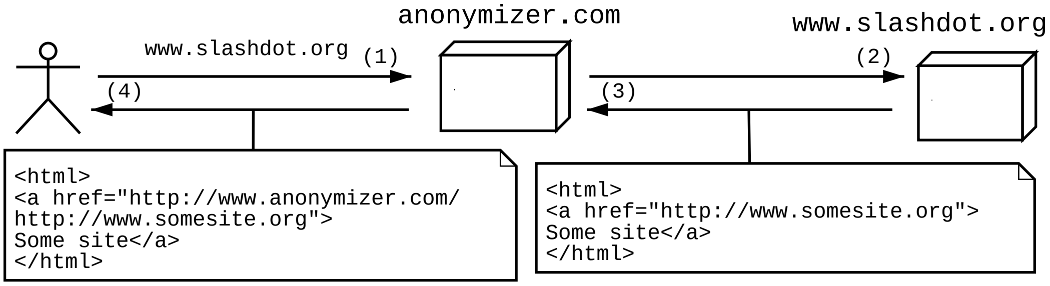
The situation has not changed much since the publication of our paper [SS03]. A user concerned about his privacy still does not have many options. In our paper, we considered the following question: what are the simplest anonymizing systems really giving a user in search of protection, and is there a way to get better protection than is given by such systems?

1.2 Anonymity and unobservability

In the terms of [PK00], *anonymity is the state of not being identifiable within a set of subjects*. *Sender anonymity* means that the sender of the message cannot be identified, whereas *recipient anonymity* means that the recipient of the message cannot be identified. *Unlinkability of sender and recipient* means that it cannot be pinpointed that a given sender and recipient are communicating with each other. Even stronger than anonymity, *unobservability* means that the existence of the message itself cannot be detected. *Sender (or recipient) unobservability* means that it is impossible to detect whether any sender (recipient) from the set of subjects is sending (receiving), whereas *relationship unobservability* means that it is impossible to detect the existence of a message sent from a possible sender to a possible recipient.

For this definition to work, one needs to consider also two concepts: that of *the at-*

tacker against whom anonymity is achieved and of *the degree of anonymity* (as described, for example, in [RR98]), the certainty with which the attacker can pinpoint the sender, the recipient, or link them to each other. The attackers can be *passive* - merely watching the traffic, or *active* - capable of inserting their own traffic. They can be *local*, watching only one node or wire, or *global*, watching multiple nodes or wires. They can watch traffic over short or long periods of time. Theoretical treatments also need to consider the computational power of the attacker.



```
<html>
<a href="http://www.anonymizer.com/
http://www.somesite.org">
Some site</a>
</html>
```

```
<html>
<a href="http://www.somesite.org">
Some site</a>
</html>
```

Figure 1.1: Single-hop proxy

1.3 Anonymizing services on the World-Wide Web

Single-hop proxy functionality.

Most anonymizing services currently available to the general public are at their base merely single-hop proxies that make the http request for the user. Figure 1.1 illustrates the operation of such a proxy. While using a single-hop proxy, the user submits the destination URL to the service, and the service immediately issues an http request to this URL. The http request appears to have originated not at the user's computer but at the proxy. As the target computer replies, sending back an html document, the proxy sends back to the user this document with all links rewritten so that they point back to the proxy, not to the sites they originally pointed to. This means that when using, for example, the free part of `anonymizer.com`, a link to `http://www.somesite.org` gets rewritten, becoming `http://anon.free.anonymizer.com/http://www.somesite.org`.

The connection of the user to the proxy may or may not be encrypted. Some services (see, for example, `the-cloak.com`) provide both; some (see, for example, `anonymizer.com`) provide only unencrypted surfing for free and charge for encrypted surfing.

Besides concealing the user's IP and possibly encryption, functionality of anonymizing single-hop proxies may include: filtering out or specially handling cookies; filtering out or rewriting JavaScript, Java, or other active content; filtering out advertisements and banners; proxying or blocking https; faking the http_user_agent field in the http header (that is, not revealing information about the user's OS and browser); faking the http_referer field (that

Name	URL	Encryption
the-Cloak	`http://www.the-cloak.com`	yes
ProxyWeb.net	`http://www.proxyweb.net`	yes
SnoopBlocker.com	`http://www.snoopblocker.com`	yes
Proxify.com	`http://proxify.com`	no
Anonymouse	`http://anonymouse.ws`	no
Guardster	`http://www.guardster.com`	paid version only

Table 1.1: Some available single-hop proxies as of December 2007

is, not disclosing the previously visited site).

Single-hop proxy examples. Table 1.1 lists a few available services. Note that most such proxies are not run by businesses that have obligation to keep them up, and therefore may appear and disappear unpredictably. Some of these services (such as, for example, `anonymizer.com`) have been subject to public review for a long time and have a reputation for not violating their users' privacy. For others, caveat emptor may apply.

Worth mention in the category of single-hop proxies is Google's translation service despite the fact that by itself it does not provide anonymity. It does rewrite links in the translated document to point at links to translations of the documents pointed to by the original links, but it does not handle links to non-text components. In the absence of non-text components, for example if the browser is configured to load only text, Google's translation service serves as a single-hop anonymizing proxy, making all the requests to the destination site instead of the user.

Single-hop proxy attacker model. A single-hop proxy attempts to protect the identity of the sender of a request from the attacker who can monitor the traffic of the destination site.

What this really means is, that leaving aside bugs and faults in design or implementation that may make such services able to disclose the information they claim to conceal (an example is in [MS02]) and leaving aside their additional content filtering capabilities, the main function of a single-hop proxy is the concealment of the user's IP from the site he or she accesses.

The destination site can do traffic analysis (see, for example, [SK02] or [Ray01]) to learn about and from the user's browsing patterns. The site can see what paths the user takes inside it (*communication pattern attack*), the intervals of time between requests (*timing attack*), and the amount of transmitted data (*packet volume and counting attack*), as discussed in [SK02]. The site may also be able to correlate different accesses by the same user that occur at different times (*intersection attack*).

Single-hop proxies also do not address the model of the attacker who may be watching user's traffic as it goes from the user to the proxy and from the proxy to the user. Sometimes the proxy does not encrypt such traffic, allowing any observer on the path the full view of what is going on.

Even if the traffic from the user to the proxy and back is encrypted, the global attacker that is able to watch both the user-proxy and the proxy-destination sides of the proxy might be able to see what data the user requests and gets by doing traffic analysis on the ingoing and outgoing traffic.

Going beyond a single-hop proxy. In 1981, Chaum introduced the concept of "Mix-nets" (see [Cha81]). "Mix-nets" are groups of servers that provide anonymity by passing user's traffic through nodes called *Mixes* which may delay, reorder, reencrypt, pad and forward traffic passing through them. In addition to providing sender anonymity, Mix-nets attempt also to provide sender and recipient unlinkability against a global attacker. Among the examples of Mix-nets are Onion Routing [SGR97], Zero Knowledge Systems' Freedom Network [GS01], Web MIXes [BFK01], Tarzan [FM02], and I2P (`http://www.i2p.net`).

Among the other approaches to the problem of anonymous web transactions is the Crowds system [RR98], which allows participating nodes to forward requests within their crowd with a certain probability. Crowds attempt to provide sender and recipient anonymity, but unlike the Mix-nets, they do not attempt to provide sender and recipient unlinkability against a global attacker.

A problem: long-term intersection attacks. A subset of intersection attacks and a major unsolved problem in anonymity systems are so-called "long-term intersection attacks" (see [BL02]), where the attacker is able to watch all or many nodes, including a number of entrance and exit points of the users' traffic. Users tend to have a certain behaviour while on-line. They tend to send messages to same or similar destinations. If user *A* in a mix daily reads a certain site, after very many observations it may be possible to match *A* with the outgoing request to that site, using the knowledge of *A*'s presence or *A*'s absence together with the information whether or not the site was visited during that time.

One of the directions for dealing with this is dummy traffic (as discussed in [BL02], [JVZ01]). Dummy traffic consists of messages sent even when the user has nothing to send. A totally different approach is proposed in PipeNet [Dai98], where the whole mix-net is supposed to shut down when one node stops to communicate with the network.

1.4 Using caching for anonymity

We propose an anonymizing web-browsing system that would deal with the global attacker who is able to do long-term observation. The proposal is to make user requests to proxy independent in time from the proxy's requests to target sites. The proxy will request and cache information on its own and send it to the users when requested. To achieve this, the proxy server would collect and store documents which its users are likely to request, provide an encrypted channel to access its contents, and rewrite links in the documents provided to the user to point back at the proxy, similarly to the workings of an ordinary single-hop proxy. This would mean that a global observer looking at the traffic to and from the server will see only regularly scheduled cache updates and encrypted user traffic, thus removing correlation between the users requests and the server's accesses to remote sites.

This caching proxy will act like a single-hop proxy, in that the destination server does not find out who communicated with it except for the proxy. But it does more than the single-hop proxy in many other respects: the destination server does not find out that someone except the caching proxy communicated with it at all, and neither does a third-party observer; the global attacker cannot correlate requests from the sender and to the recipient; and long-term intersection attacks are meaningless since there is no connection between the

	Sender anonymity from		**Recipient anonymity from**		**Sender, recipient unlinkability**	**Single point of failure**
	recipient	**3rd party**	**sender**	**3rd party**		
Single-hop proxies	yes	no	no	no	no	yes
Mix-nets	yes	yes	no	yes	short-term	no
Crowds	yes	yes	no	yes	no	no
Unencrypted caching[1]	yes	no	no	no	no	yes
Encrypted caching[1]	yes	yes	no	yes	yes	yes

Table 1.2: Anonymizing systems compared

proxy's fetches and the user's requests.

In short, the attacker model includes not only the attacker at the destination site (as that of existing single-hop proxies) but also the global attacker who is able to do not only short-term, but also long-term attacks.

Similarly to the existing single-hop proxies, the user will not be protected from the attacker at the proxy itself.

Such a proxy would compare with the existing anonymity services as described in Table 1.2.

1.5 Similar work and candidates for a prototype

Caching proxies that periodically update their content. The best known example of a caching proxy that collects data on its own for future use is Google's cache [Goo]. Every few weeks it crawls all the web, collecting, storing and indexing text information from every webpage [BP98]. Attempting to scale to the size of the whole World Wide Web, it has been optimized for memory usage and speed.

Google's primary objective is indexing text information. It does make the attempt to index images but only caches thumbnail images, relying on the original image being still

[1] Proposed for providing anonymity in this paper.

present on the server. Such usage may be due also to data size considerations, but there are also legal reasons for it. In a recent case, the 9th U.S. Circuit Court of Appeals ruled that it is legal for search engines to cache thumbnail images [Ols03a], but it is yet to be determined whether it is legal to cache a full-size image copy. According to the Court, the use of thumbnail images is fair use, because they “do not supplant the need for originals” and “benefit the public by enhancing information-gathering techniques on the Internet”.

Other search engines that allow access to their caches are Comet Web Search [Com] [2] and Gigablast [Gig]. An interesting web cache is Wayback Machine [Way]. This is an internet archive that contains multiple copies of webpages from 1997 to 2002. Similarly to the caches of search engines, it cannot be directly used for anonymous browsing. Using JavaScript, Wayback Machine handles links to point back to the cache where archived versions are available, but if that fails, it sends the request to the original site.

“Using Google cache as anonymous surf”. The idea of using Google cache for anonymity has occurred to Google’s users before, but Google’s design is not targetted towards this goal. Written apparently as a result of such attempts, an FAQ entitled ’Don’t Use the Google Cache as Anonymous Surf’ [The03] explains that clicking on a link in a cached page would lead you to a page outside Google cache, an embedded image would attempt to load directly from the original site, and redirection might make the whole page load from the original site. In order to use Google cache for anonymity, our scheme eliminates all outgoing links.

1.6 Prototype

We prototyped[3] an anonymizing caching proxy that outsources all caching to Google by forwarding all requests to Google cache and rewriting links in the resulting documents. As shown in Figure 1.2, when a URL is submitted to the script, the script requests a cached

[2] Comet Web Search stopped allowing access to their cache sometime after the submission of this article. The reasons for this are unknown to the authors.

[3] The prototype is at `http://www.cs.dartmouth.edu/~ashubina/google.html`. It is a CGI script that can be located at the user’s machine or elsewhere.

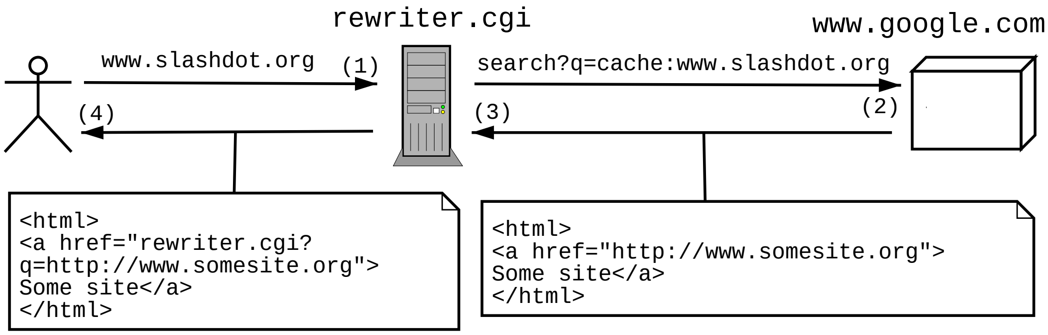


Figure 1.2: Anonymizing caching proxy

copy from the Google cache. Any link in the document gets rewritten to submit the link to this script, instead of going directly to the requested page. Thus, if we deal with plain text data, every request would be redirected to the Google cache.

The implementation has to detect all outward going links, detect their data type and to rewrite them according to whether their content may be available from Google cache.

Since Google cache does not store images, images are not provided unless the user explicitly specifies that he or she wants the script to act like a single-hop proxy that fetches images for him from the destination server. Such a transaction would make the destination server aware that somebody is accessing the content and disclose the user's browsing to a global observer. If the script is located on the user's machine, it would also disclose his IP to the destination site, whereas if the script is not located at the user's machine, this behaviour would be no different from that of a single-hop proxy.

1.7 Performance

The speed of browsing through a caching proxy as compared to the speed of browsing the original website is dependent on a number of parameters, including the routes to the original server and to the proxy, the bandwidth of the corresponding connections, and the speed of accessing files in the cache.

To reduce the contribution of some of these factors, we chose pages hosted by Google itself for our study on the performance of Google cache. We ran tests downloading various

	From original server	**From cache server**
Average page load time	0.525 sec	0.483 sec
Min page load time	0.280 sec	0.420 sec
Max page load time	0.930 sec	1.000 sec

Table 1.3: Download times, 12K page

	From original server	**From cache server**
Average page load time	0.478 sec	0.457 sec
Min page load time	0.320 sec	0.430 sec
Max page load time	0.770 sec	0.500 sec

Table 1.4: Download times, 16K page

pages hosted by googlepages.com directly from googlepages.com (Google-owned webpage hosting) and from the Google cache. In all the cases, the performance penalty incurred by accessing the cached version instead of the original was small enough to be unnoticeable to a human.

The results of our experiments can be found in tables 1.4–1.7 below. Page sizes varied from apparently typical range of 10K–40K to the unusual 200K. Cached page versions were about 2K larger than the originals, due to the extra header added by the Google proxy. It should be noted that accesses to the cached page were slowed down by redirects to the caching server from google.com. Our testing scripts and more information on the choice of target pages can be found at `http://www.cs.dartmouth.edu/~ashubina/googlepages.tgz`.

The above tables for pages of different sizes are summarized in Fig. 1.3. Notably, cached page download times exhibited less variation than the respective original pages, being served from a different groups of servers.

We conclude that, given a sufficiently powerful caching proxy, accessing a cached version of a page instead of the original version does not have to cause a noticeable performance hit.

	From original server	From cache server
Average page load time	0.569 sec	0.533 sec
Min page load time	0.410 sec	0.460 sec
Max page load time	1.140 sec	1.040 sec

Table 1.5: Download times, 27K page

	From original server	From cache server
Average page load time	1.145 sec	1.254 sec
Min page load time	0.843 sec	0.913 sec
Max page load time	1.504 sec	1.425 sec

Table 1.6: Download times, 36K page

	From original server	From cache server
Average page load time	1.070 sec	1.473 sec
Min page load time	0.930 sec	1.000 sec
Max page load time	1.420 sec	4.310 sec

Table 1.7: Download times, 198K page

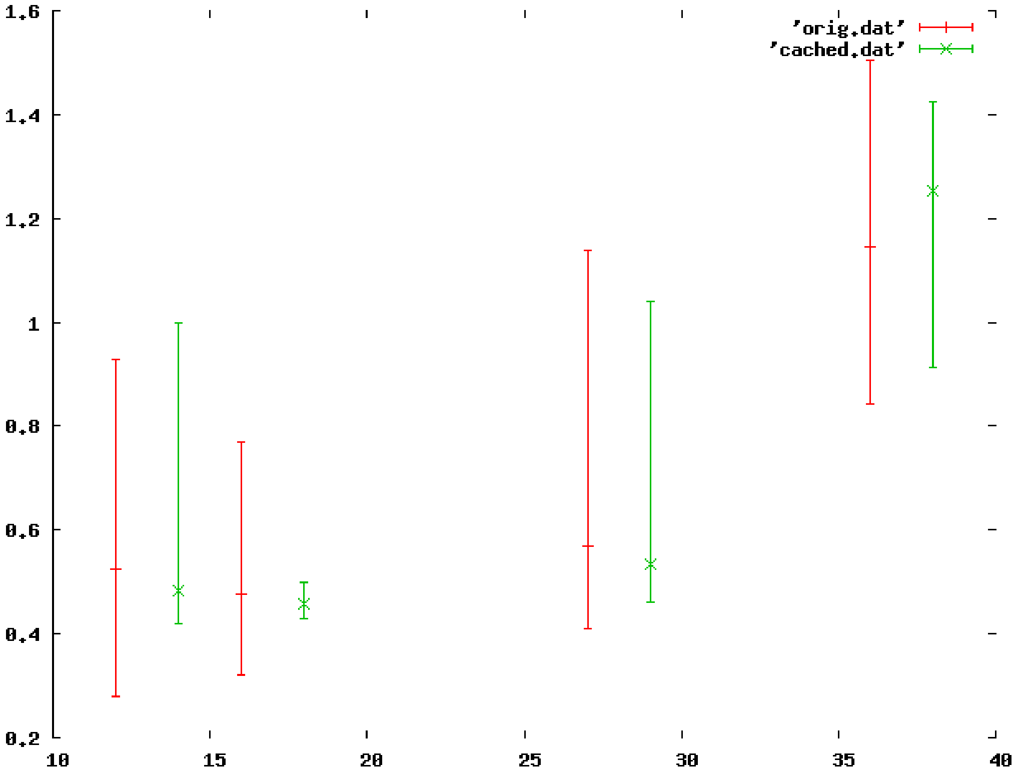


Figure 1.3: Download times for original and cache servers, by page size (KBytes)

1.8 The caching proxy controversy

It appears to be an unfortunate law of the modern civilization that whenever anyone wants to give out something for free, someone else is going to have a problem with it. Caching proxies are no exception to this rule.

Most Web sites strive for high search engine rankings (on how to play this game, see, for example, [CD03]). However, as described in a recent `news.com` article [Ols03b], many of them are unhappy with caching of their contents.

[Ols03b] gives the following list of complaints about search engine caching:

- **Caching of removed data.** Caching lets users access pages that are temporarily unavailable. Thus it sometimes provides also access to content that has been deliberately taken off-line.

- **Access to registration-only sites.** Due to faults in the design of a number of sites that require registration, caching may end up indexing and caching their content.

- **Detouring traffic from the original sites.** The very feature that we propose to use for anonymity is a subject of such complaints. The claim is that by detouring traffic from the sites where the original information is stored, caching may make Web publishers lose their income.

Interestingly, for the current implementations of all the web caches we reviewed ([Goo], [Way], [Com], [Gig]), the last claim is not quite valid. Among the many ways in which the user's anonymity from the original site provided by the use of such caches is violated are images, which usually include also ads. Since ads are not cached, a cached copy of a webpage with an ad usually displays the ad downloaded from its original provider. Such ads may, however, not be able to track to which site they really owe this request very well, and thus may mistake the web cache for the referrer. (However, the proxy we prototyped has a different behaviour.)

Google's response to the last complaint is that web sites can easily prevent Google from caching their pages without preventing it from indexing them. All it takes is adding to your page either <`META NAME="ROBOTS" CONTENT="NOARCHIVE"`> to exclude all robots, or <`META NAME="GOOGLEBOT" CONTENT="NOARCHIVE"`> to exclude just Google. However, some sites are reluctant to do this for fear that this may affect their search rankings.

The Digital Millennium Copyright Act has a narrow exception for Web caching, allowing internet service providers to keep local copies of Web pages. It is not clear, however, whether this would protect also search engine caches or archives similar to [Way] if tested.

The legality of a caching proxy such as we prototyped under the modern copyright laws is yet to be determined, as is the legality of search engine caches. Due to the very features that make our scheme provide anonymity that are not there for other web caches, our scheme may not be ruled legal even if other web caches are.

1.9 Extensions

Obviously, a proxy such as we prototyped would not allow the users to use any dynamic content. A user could not provide any information to be passed to the destination site without informing the destination site and the global observer that someone is using the proxy to look at the destination site. It might be possible to design a proxy that submits such a request for the user together with other people's requests or a number of randomly generated requests, but such an extension is bound to be subject to attacks.

However, a natural extension would be allowing to search Google cache, using Google itself and rewriting the results to point back into the cache. It might also be possible to design a caching proxy that provides thumbnail images, as does Google.

Google (as well as the other caching proxies described above) does not provide encryption, hence the information going to and from Google is not encrypted. An encrypted proxy would allow protection from an attacker that can watch traffic going to it. Against such an attacker, it would also help to bundle packets together or break them up, normalizing their size. Such a proxy could also attempt to conceal its operations from an attacker at the very proxy by means of private information retrieval.

1.10 The future of caching proxies

In this chapter we presented an approach that would allow retrieving text information from the web with better anonymity than a single-hop proxy allows to achieve and would also eliminate long-term correlation attacks by the global attacker. The approach uses a caching proxy that contains its own copy of the World Wide Web (or of its large subset). One example of such a proxy is Google cache, which we used for a prototype.

Our prototype is already useful both for achieving limited anonymity, and for accessing cached copies of sites that for some reason are unaccessible at the moment.

As privacy of web transactions becomes more and more important and harder and harder to achieve, it may be worthwhile to use caching proxies to achieve anonymous access to what may be the most important achievement of the human civilization - written text information.

Chapter 2

A coercion-resistant, voter verifiable electronic voting system

The problem of producing a fair voting system is well-known and, as demonstrated, for example, by the US presidential elections of 2000, not a simple one to solve. At the very least, such a system should protect voters from coercion by adversaries; it is also highly desirable, and not usually achieved in practice, that voters be able to verify that the tally reflects the sum of the votes that were actually cast, as they were intended to be cast. Currently most proposed voting systems fall short in this regard: they either do not provide both coercion-resistance and verifiability or require the voter to be a computer.

In the paper "Design and Prototype of a Coercion-Resistant, Voter Verifiable Electronic Voting System" (A.M. Shubina and S.W. Smith, 2004) we discuss the requirements of a fair voting system and propose a system that satisfies these requirements.

The following requirements are generally considered to be desirable for an electronic voting system.

1. *Correctness.* Every voter should be able to vote but only once; only votes cast by registered voters should be included in the tally; all votes should be correctly counted.

2. *Privacy.* Nobody except the voter can find out what choices the voter made without

interacting with the voter.

3. *Receipt-freeness or coercion resistance.* Receipt-freeness means that the voter cannot carry away any evidence of how he voted. The property of coercion resistance is stronger: it means that the voter should be able to cheat an adversary who may interact with him and instruct him to vote in a given manner.

A desirable property is also that of *verifiability*. Verifiability would allow the voter to check that his vote was committed as intended and made it to the final tally as cast (*voter verifiability*) and would also allow any observer to verify the tally (*universal verifiability*).

Intuitively, the property of voter verifiability appears to be incompatible with the property of receipt-freeness: if the voter is able to verify that his vote was counted as he cast it, what can prevent him from proving how he voted to a third party? This seeming incompatibility has been addressed in literature by cryptographical means. The voter may be given an encrypted copy of his vote and allowed to verify that his vote made it to the final tally. However, this either requires the voter to be able to verify that the encryption is correct or makes other sacrifices: for example, in case of Chaum's layered receipts, the voter can be cheated with probability 50%.

We produced an electronic voting scheme that sacrifices a certain amount of privacy but allows nearly perfect voter verifiability. Our scheme addresses the contradiction between verifiability and receipt-freeness in the following manner: we rely on the information in the voter's head, known to be true to the voter but impossible to prove to be true to anybody else. The idea is to allow the voter to carry away a receipt that is uninformative for a coercer without access to the voting machine or to the contents of the cast ballots. The receipt would point to a number of vote records in the central database, one of which, as remembered by the voter, is the correct record of the voter's vote, whereas the others point at different records. Our system does not assume any trust in the voting machine, but requires a few other assumptions, which we believe to be reasonable in the real-world situation. A basic prototype of this system is available on our website at `http://althing.cs.dartmouth.edu/cgi-bin/electme2/master.pl`.

Section 2.2 briefly discusses the requirements for a secure election system. Section 2.3 presents a brief overview of methods used in receipt-free election schemes. Section 2.4 discusses the most recently implemented election schemes. Section 2.5 presents our election scheme. Section 2.6 discusses the practical applicability of our scheme. Section 2.7 describes our prototype. Section 2.8 offers some concluding remarks.

2.1 Solving the problem of fair elections

The US presidential elections of 2000 made the general public aware of the problems of producing a voting system that could be trusted by the voters to submit their votes correctly. Despite the public review and control of the US electoral system, many US citizens felt that the system had failed them. Although the problems did not originate in the 2000 election, the situation where a very small number of votes was sufficient to flip the final tally raised the public's awareness of the inadequacy of the system.

The problem of producing a fair voting system has been well-known in countries and situations where adversaries have a very high degree of control. In totalitarian societies (or other situations with almost complete adversarial control), it may be futile to attempt to solve this problem. Such societies provide no guarantee that the adversary will comply with the solution, no guarantee that the observers and complainants will be able to speak up, and no guarantee that the situation will be corrected even if there is a valid complaint. However, in a free democratic society in the 21st century, the electoral system is subject to public review and control. Its failures do not have to be possible.

If an electronic voting system is to be applied in secret-ballot elections, it has to be *receipt-free*, i.e. not allow a voter to carry away any evidence of who he voted for, since such evidence would permit vote buying and coercion. Receipt-freeness is hard to combine with *voter verifiability*: if a voter is able to verify that his vote was counted as he cast it, what could prevent him from proving how he voted to a third party?

Cryptography can help address this seeming incompatibility between receipt-freeness and voter verifiability, for example by allowing the voter to carry away an encrypted copy of his vote and to verify that his encrypted vote made it to the final tally. However, that requires the voter to be able to verify that the encryption of his vote is correct. Chaum's layered receipts [Cha04] (discussed below) solve this problem partially: they allow verification, but only with probability 50%.

In this paper, we examine the desirable properties of electronic voting systems and survey the principal current approaches to such systems. We then present a new design that improves on the previous work, by being (arguably) the first one that achieves both voter verifiability and coercion resistance, while not assuming the voter is a computer, not relying on correct behavior by the voting machine, and detecting close to 100% of misbehaviour.

2.2 Requirements for an election system

The two basic properties commonly required of election systems are *correctness* and *privacy*. An election system should produce a correct *tally*—the result of the election. It should also ensure that the participant's vote will remain *private* (infeasible to find out without cooperation with the voter)—although sacrificing the privacy requirement permits making the system much simpler.

The more interesting properties extensively studied in theoretical literature are *receipt-freeness* and *coercion resistance* (as described in Section 2.2.3) and *verifiability* (as described in Section 2.2.4).

Would ensuring correctness, privacy, receipt-freeness, and verifiability be sufficient for a real-world election system? Arguably, almost all problems in existing real-world election systems stem from the lack of one of these properties. Less discussed, and not ensured by any of these properties, is another important property: *trust*.

2.2.1 Correctness

The very first requirements of every voting scheme are that every voter should be able to vote, but only vote once; only votes cast by registered voters should be included in the final tally; and all votes cast by registered voters should be correctly counted.

Defining a correct tally is somewhat harder, because it is unclear how to handle incorrectly cast votes. In the definitions of Benaloh and Tuinstra [BT94], a tally is *correct* if correctly cast votes representing a valid choice are counted, whereas incorrectly cast votes that do not represent a valid choice may be counted one way or another. It may be worth noting that in a real-world situation it may be unacceptable to include incorrectly cast votes into the tally, and thus the ability to distinguish incorrectly cast votes may be important. Hirt and Sako [HS00] get around this problem by requiring for correctness that no voter be able to cast an invalid vote. Other possible approaches could be allowing to either discard or correctly fix invalid votes.

2.2.2 Privacy

Privacy, in the context of elections, means ensuring that nobody except the voter can find what choices the voter made without interacting with the voter. More precisely, an adversary should not obtain more information about a voter's vote than provided by the election tally.

Privacy of a voting system is dependent on assumptions as to what the adversary can or cannot do and is often based on assertions about the adversary's computational power.

One of the possible physical assumptions made for privacy is a *voting booth* that allows the voter to secretly and interactively communicate with an authority. A weaker assumption is an *untappable channel* that allows a voter to send a message that cannot be observed by the outsiders. Whereas voting booths are used for voting in the real world, they are typically used only for communication with the voting machine and do not allow remote communication.

2.2.3 Receipt-freeness and coercion resistance

The initial papers on secret-ballot elections considered the property of *receipt-freeness*. As introduced by Benaloh and Tuinstra [BT94], receipt-freeness is the inability of a voter to prove to an adversary how he voted, even if the voter would like to provide this proof. Receipt-freeness is necessary in secret ballot elections. Indeed, if the voter had the ability to prove to an adversary the contents of his vote, the adversary would be able to demand from the voter that he vote in a particular manner and reward him for voting in this manner or penalize him for not complying with the demand.

More recently, Juels and Jakobsson [JJ] introduced a stronger notion of *coercion resistance*. A coercion-resistant system is a system where the voter can cheat an adversary who may interact with him and instruct him to vote in a given manner, but the adversary will not be able to determine whether the voter behaved as instructed—even if the adversary asks the voter to disclose his keys or to abstain from voting.

2.2.4 Verifiability

In the ideal voting scheme, the voter should be able to verify that his vote was committed as intended and made it into the final tally as cast (*individual verifiability*, or *voter verifiability*), and any observer should be able to verify the tally (*universal verifiability*).

These two properties provide *verifiability*—the possibility of verification that all votes are counted correctly.

2.2.5 Trust

The question of trust in a voting system is one of the most discussed and least agreed upon. Who should trust whom for what? Is it enough for an expert to trust another expert's assertion that the system functions correctly?

The range of opinions on what would constitute a trustworthy electronic voting system is very wide, ranging from Rebecca Mercuri's *Statement on Electronic Voting* [Mer01] that demands the use of "an indisputable paper ballot" and the Mercuri Method [Mer02], to Andrew Neff's [NA03] and David Chaum's [Cha04] reliance on verification. There is, as yet, no agreement on whether it would be sufficient to have a system that the experts can prove sufficiently untamperable.

2.3 Basic cryptographic schemes

The following cryptographic approaches for secure electronic voting have been widely discussed in literature.

2.3.1 Voting schemes based on homomorphic encryption

Homomorphic encryption is encryption over an algebraic group such that the encryption of the sum of two elements of the group is the sum of the encryptions of these elements. The idea of using homomorphic encryption in electronic voting is to sum the encrypted votes, and then decrypt the sum—without decrypting individual votes.

The first homomorphic encryption voting scheme was proposed in the paper by Benaloh and Tuinstra [BT94]. This scheme was proven by Hirt and Sako in [HS00] not to be receipt-free. In the same paper Hirt and Sako proposed another, receipt-free, homomorphic encryption voting scheme.

Homomorphic encryption schemes do not support write-in votes.

2.3.2 Voting schemes based on mix-nets

A number of electronic voting schemes are based on Chaum's *mix-nets* [Cha81]. Mix-nets encrypt, permute, and re-encrypt the sequences of input elements, producing permutations of these elements intended to conceal their original order.

In mix-net voting schemes, a vote is encrypted with a sequence of the keys of the authorities and consequently decrypted by the authorities who prove the correctness of the decryption. An example of such a receipt-free voting scheme is the scheme proposed by Sako and Kilian in [SK95].

Mix-net based schemes can support write-in votes.

2.3.3 Voting schemes based on blind signatures

Blind signatures (also due to Chaum) allow an authority to sign an encrypted message without seeing its contents. In electronic voting schemes, blind signatures are used to allow the administrator to authenticate a voter by signing an encrypted ballot.

An example of a scheme based on blind signatures is the scheme proposed by Okamoto in [Oka96]. Okamoto later showed this scheme not to be receipt-free and fixed it in [Oka97].

Schemes based on blind signatures can support write-in votes.

2.3.4 Voting schemes based on ring signatures

Ring signatures, first introduced by Rivest et al. in [RST01], are signatures that do not allow to determine the identity of the member of the group of signers who signed the message. In e-voting applications, ring signatures are used to allow voters to sign their votes without leaking their identity. To avoid double voting, *linkable ring signatures* – signatures that allow to determine whether two messages have been signed by the same signer – have been used in [LWW04], [LW05], and [TW05].

2.4 Some notable election schemes

Recent literature also provides examples of practical, implemented schemes.

2.4.1 David Chaum's encrypted receipts

David Chaum's new scheme [Cha04] allows a voter to walk away from the polling place with an encrypted receipt that has previously been shown to him to correctly contain his vote.

The scheme functions in the following way. The voter chooses his votes electronically. The voting machine prints out a two-layer image that displays the vote. The user selects which part of the image—the top layer or the bottom layer—he would like to keep, and walks away with it. The voter can later verify that the receipt was correctly posted on the election site by looking it up by the receipt's serial number in the *receipt batch*, the set of receipts the authority intends to count. The *tally batch*, the set of plaintext images of ballots as seen in the voting booth, is also posted in random order, allowing everyone to verify the tally.

On each ballot, the voting machine can cheat with probability 50% by either printing one incorrect layer, or reusing the serial number, or performing a tally process step incorrectly. Thus, if even only 10 modified ballots are verified, the chances that the tampering will go undetected would be less than 1 in 1000.

The scheme does not answer the question of what to do if the election did get tampered with. Since only 50% modified ballots will be detected, there is no opportunity to recast only the modified ballots.

2.4.2 VoteHere

In an attempt at providing voter verification and ensuring user trust, the VoteHere [Vot] [NA03] system hands a voter a paper receipt.

In the VoteHere scheme, a voter starts with a *voting token* (a smart card or a key). This voting token can be inserted into a machine that lists all candidates and provides a *verification code* for every candidate. The verification codes are different for every voting token. The voter picks the codes corresponding to his choices and gets a receipt listing the ballot number, these codes, and the signature of the ballot produced by the voting machine. Later, the voter can use the ballot number to verify that the codes were recorded correctly; however, he cannot verify that the voting machine did not swap candidates before presenting to him the codes. Instead, the voter should trust that the trustees responsible for the generation of codebooks and auditing of the machines made this impossible.

2.4.3 Rivest's ThreeBallot, VAV, and Twin systems

ThreeBallot

In Rivest's ThreeBallot system [Riv06] the ballot consists of three ballots, which are originally identical except for their numbers. Each candidate has an empty circle corresponding to him on every ballot. The voter is supposed to vote for a candidate by filling in 2 of the 3 circles and against a candidate by filling in only 1 circle.

BALLOT		BALLOT		BALLOT	
President		President		President	
Alex Jones	○	Alex Jones	○	Alex Jones	●
Bob Smith	●	Bob Smith	●	Bob Smith	○
Carol Wu	○	Carol Wu	●	Carol Wu	○
Senator		Senator		Senator	
Dave Yip	●	Dave Yip	○	Dave Yip	○
Ed Zinn	○	Ed Zinn	●	Ed Zinn	●
3147524		6808952		5593816	

The checker machine verifies that the ballot is valid. The voter selects one of the three parts of the ballot and carries away a copy of it. The three ballots are separated and submitted. All ballots get published, and each voter is able to verify that his receipt matches a published ballot.

VAV

Proposed in [RS07], Rivest's VAV (Vote / Anti-Vote / Vote) system also requires the voter to fill out three ballots, two of which are marked "V" for Vote, and one is marked "A" for Anti-Vote. The Anti-Vote ballot and one of the Vote ballots (as selected by the voter) negate each other and must be identical except for the "A" and "V" indicators. VAV can support any kind of voting system. In the example below, the VAV system is used in a ranking scheme, and the last two ballots negate each other.

BALLOT		BALLOT		BALLOT	
V		A		V	
Alice	1	Alice	2	Alice	2
Bob	2	Bob	1	Bob	1
Mallory	3	Mallory	3	Mallory	3
5077332		9556124		3832472	

Similarly to the ThreeBallot scheme, the voter may carry away a copy of any of the three ballots.

Twin

The voting system called “Twin” was introduced by Rivest in [RS07]. In this scheme, the voter fills out just one ballot, which has its number hidden under a scratch-off cover. The scratch-off cover is removed after the ballot is submitted. The voter gets to take home a copy of a random previously-cast ballot.

2.5 Our voter-verifiable election scheme

All of the above described voting schemes—both the theoretical ones and the implemented ones—appear to allow receipt-free implementations. However, verifiability (and also correctness, in schemes relying on verifiability for correctness) turns out to be harder to achieve in practice than in theory. Here are just a few problems with the schemes listed above. VoteHere’s scheme depends for its correctness and verifiability on the correct behavior of the voting machine. Chaum’s scheme detects voting machine misbehavior in only 50% of votes. In Rivest’s ThreeBallot and VAV schemes, if one part of the ballot is tampered with, the chance that the voter will find this out is only $1/3$; also, a corrupt checker may allow voters to overvote. In Rivest’s Twin scheme, a voter cannot verify his or her own ballot, and therefore a small political party may feel that not enough of ballots of its members get verified by its other members.

To make a step toward ensuring users’ trust, we would like to propose a scheme that would allow a user to verify how he voted, not only that a vote in his name made it to the destination. Our scheme does not make any assumptions about the correct behavior of a voting machine and makes voting machine misbehavior detectable in almost 100% cases, achieving correctness and voter verifiability at the cost of one not commonly made assumption: that the initial process of generating tickets leaks no information.

This goal is hard to achieve without sacrificing coercion resistance; indeed, if the voter can verify how he voted, then what prevents him from proving to someone else how he voted? We will attempt to solve this problem by taking into account voter knowledge—that is, information known to the voter but not accessible to the coercer.

In our scheme, we make an attempt to achieve voter trust by permitting a voter to verify his vote by using a printed receipt to query the central authority for the record on a certain key. However, in order to avoid coercion, the voter should possess also the keys corresponding to his other possible votes and should be aware of this correspondence, perhaps by having it recorded on the same printed receipt. Also, in order for this idea to work, the voter's choice should be specified by the voter in such a manner that no malicious authority would be able to tamper with it without detection.

Our scheme provides the voter with pre-generated keys and allows him to permute them, matching the keys with the possible votes. This permutation ensures that a malicious authority cannot on its own generate a desired ballot. The permutation is also used to provide a visually satisfying receipt and verification routine. The voter's choice is specified by submitting the key corresponding to the desired candidate. Finally, a signature of the ballot is generated by the voting machine. This signature is printed on the voter's receipt and publicly posted. The signature can be used by the trusted observers to check that the recorded ballot matches it.

2.5.1 Assumptions

In our scenario, we are trying to imitate the real-world model of electronic voting. We assume that the world consists of:

- the central authority A (in the real world, the central election committee);
- local authorities $B_1, \dots, B_m$ (in the real world, the voting machines);
- n voters $V_1, \dots, V_n$;
- only a finite number d of possible choices $c_1, \dots, c_d$ for all votes.

(One of these possible choices should be "no selection", to allow the voter to abstain from voting. A few choices may be dummy votes.)

We assume no communication by the central authority with the voters, but we assume the existence of voting booths—that is, communication of voters with the voting machines. This latter assumption does not appear to be unreasonable; voting booths are frequently used in the real world for ensuring voters' privacy.

Our scheme involves tickets generated by the central authority A. We require another assumption: that no information leaks from the initial process of generating voting tickets leaks to local authorities. One method of ensuring this assumption could be having different authorities participate in generating keys on cards, so that no authority would have the full information.

2.5.2 Process

The steps necessary to cast a vote are illustrated in Figure 2.1.

- The central authority A (or a set of authorities) uses its public-key encryption function E to encrypt some numbers. (This encryption function should not allow an adversary to guess the plaintext and then verify this guess by encrypting it. It would suffice, for example, to use RSA encryption with OAEP ([BR94]) padding.) The authority uses E to encrypt numbers $1, \ldots, n$ to serve as ticket identifiers. (We use n, so there will be a distinct identifier for each voter.) The authority also encrypts numbers $1, \ldots, dn$ to serve as keys printed on tickets. (We use dn, so there will be a distinct identifier for each choice, for each ticket.)

  For each ticket i, let us define $F_i(k) = E(di + k)$ (that is, the kth key on the ith ticket). The authority stores both the ticket identifiers and the keys.

- The central authority A then prints out n tickets (but does not publicize their contents). Ticket i consists of the ticket identifier $E(i)$ and keys $F_i(1), \ldots, F_i(d)$. In the real world the authority could either produce a number of sealed envelopes containing these tickets, or make them obtainable from the official election website only.

- Each voter selects a random ticket (for example, by pulling it out of a box or by

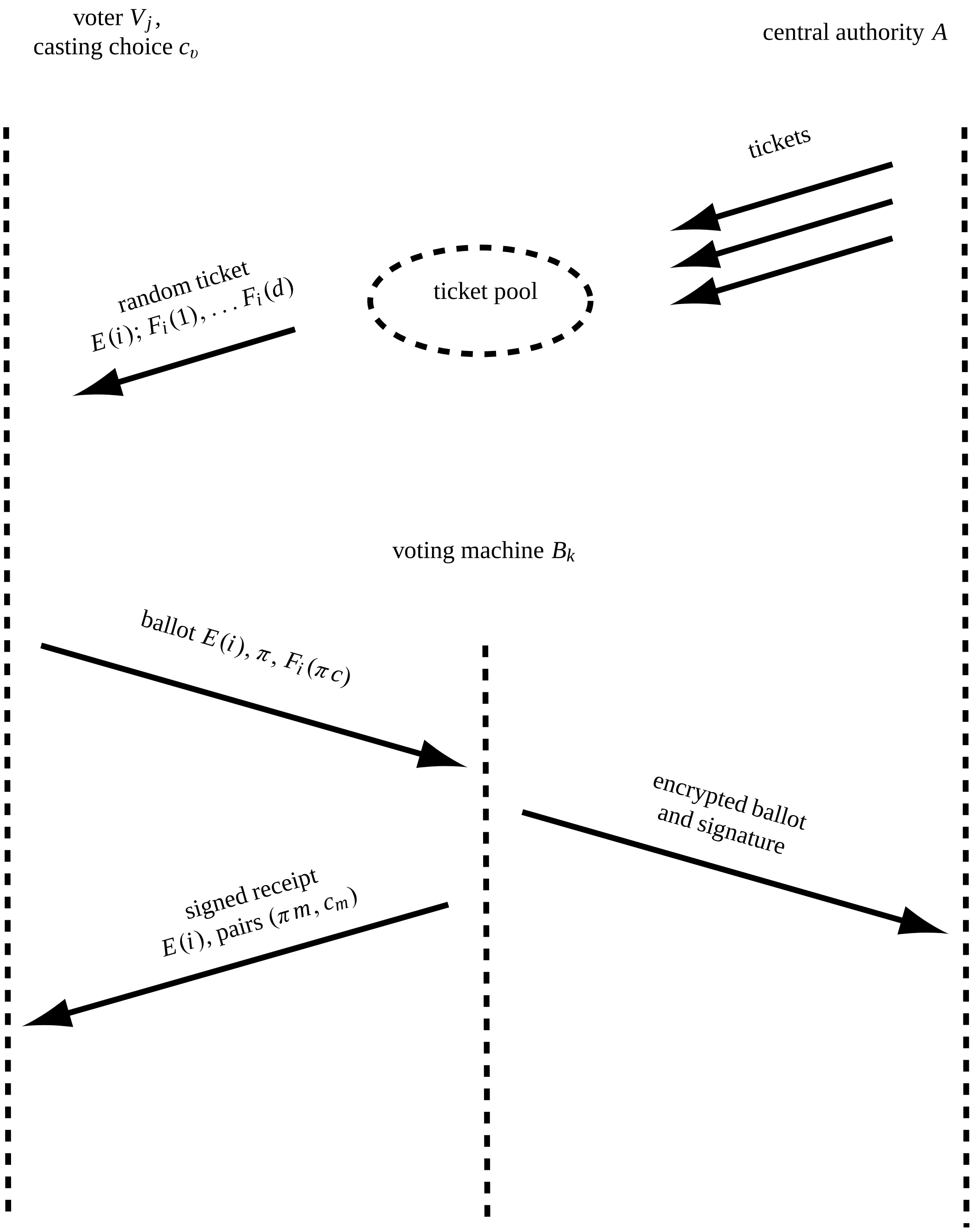


Figure 2.1: The voting process

getting it from the election website).

- Suppose voter V_j selected ticket i and wishes to vote for choice c_υ (for some υ with $1 \leq \upsilon \leq d$). Then V_j casts to his local authority B_k a ballot consisting of
  - ticket identifier $E(i)$;
  - a random permutation π of $1, \ldots, d$;
  - $F_i(\pi\upsilon)$. (Recall that υ is the index associated with voter's choice c_υ.)
- The local authority B_k prints out a receipt including ticket number i, all pairs $(\pi m, c_m)$ (ordered by πm) and the signature of the ballot, as shown in Table 2.1. B_k hands the receipt to the voter.
- B_k encrypts the ballot and signature with the public key of the central authority and submits this to the central authority A.
- When the central authority A receives this ballot, it checks that it is properly formatted—and that the key (allegedly $F_i(\pi\upsilon)$) really does decrypt to the index of a valid choice c_υ. The authority A also verifies that the signature provided is valid for this ballot. If both these conditions are satisfied, the authority A records this ballot as a vote for c_υ.

Within the ballot, the only information about the voter's selection is $F_i(\pi\upsilon)$, the key corresponding to the permuted index of the voter's real vote. The central authority A can decode it and find $\pi\upsilon$, thus finding the real vote (unless corrupted by the local authority). However, by the assumptions of cryptography, the local authority cannot extract $\pi\upsilon$ from the key, and thus cannot find out who the voter voted for (unless the central authority A cheats).

Note also that the voter does not submit keys $F_i(t)$ for $t \neq \upsilon$. Thus if the central authority did not share information with the local authority, the local authority cannot know what these keys are.

If the local authority does not know the other keys on the ticket, it cannot manufacture a ticket that features the votes in the same order but with an incorrect vote matched to a real

Table 2.1: An example voter receipt

Ballot ID	3708DD567880145B
Index	Vote
1	
2	no selection
3	Bob
4	
5	
6	
7	
8	Alice
9	
10	
Signature	0DA7E8339A56730024C

key. The local authority can attempt to record a different vote under a correct key or corrupt the correct key, but this attempt will later be caught by the verification process.

After the election is done, the central authority A publishes the receipt signatures, thus allowing any voter to verify that his ballot made it to the final count. For additional assurance, the authority A can provide a post-election query service: a voter can submit any key $F_i(\pi t)$ (and the ticket identifier $E(i)$), and check that he gets back c_t as shown. Figure 2.2 shows this process. If he sees that his vote did not make it as intended, he can request resubmission.

2.5.3 Properties

The ability to look a vote up by the corresponding key allows voter verification. Indeed, suppose looking up all keys on the ticket matches the votes submitted by the voter under the corresponding indices. This could only be achieved in three ways.

1. The real vote is submitted correctly.

2. The local authority submitted a fake vote as a real vote. However, this would require that the local authority know a key different than the one the voter submitted. Since the local authority does not know any of the other keys printed on the voter's ticket, it cannot swap the real and the fake vote provided by the voter without being detected.

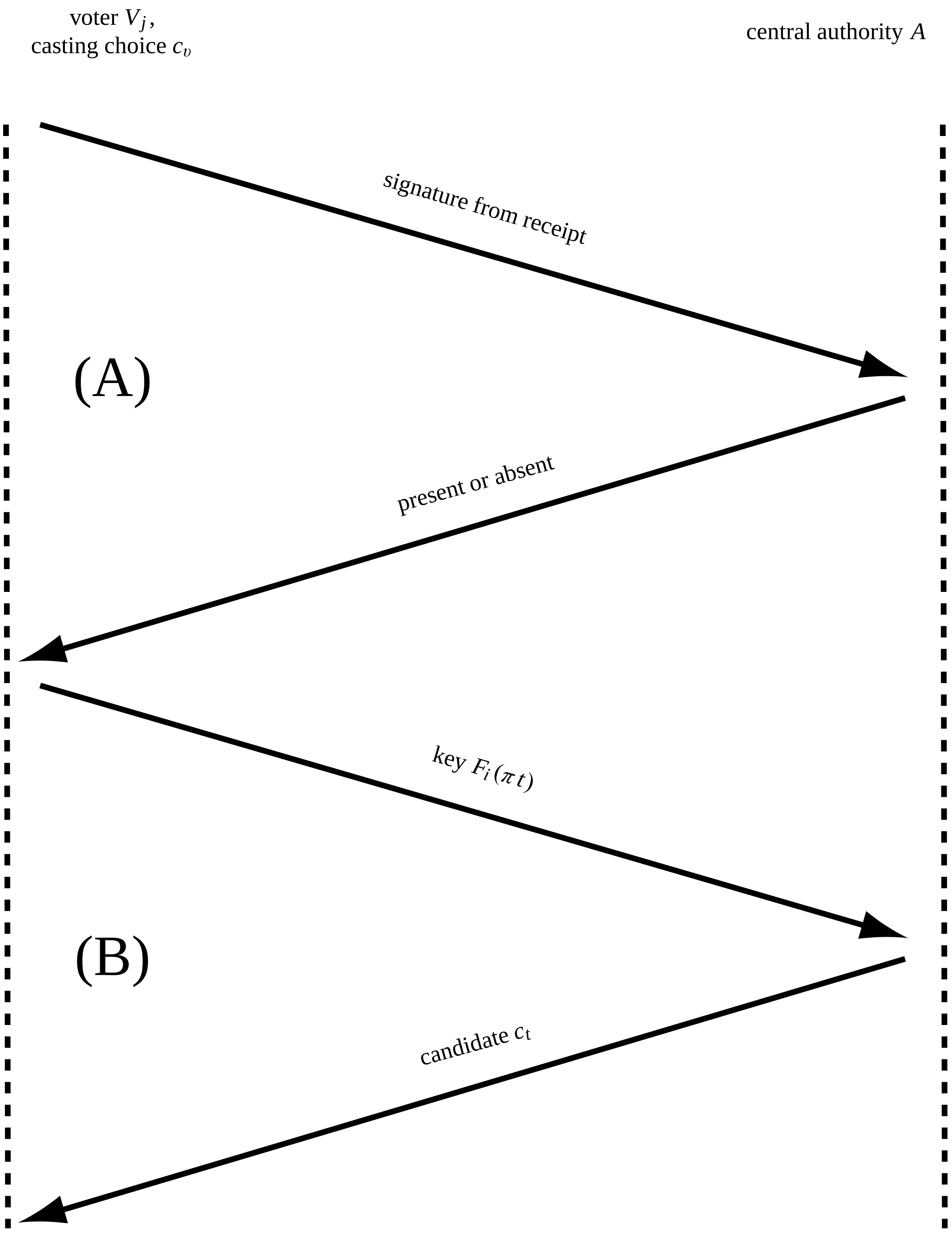


Figure 2.2: The verification process, for voter V_j with ticket $E(i)$. In (A), a voter checks that his ballot made it; in (B), a voter checks that the ballot that made it in matches his ticket.

3. The central authority considers a fake vote to be a real vote. However, the signatures are published and can be checked by voters, and trusted parties could verify that the decryptions of the encrypted votes contain signatures matching these records.

Therefore, if the local authority does not have any prior knowledge of the keys, neither the local nor the central authority can tamper with the voter's choice without detection. Thus the voter can verify whether his vote is submitted correctly.

As mentioned above, the central authority cannot lie about the contents of the ballots it got, because signatures will be posted and can be checked by voters, whereas the fact that the decryptions got correctly counted may be verified by trusted parties.

If an external adversary who does not have access to the voting machine or to the central authority's data instructs a voter to behave in a certain manner, the only evidence of the voter's behavior will be his receipt (unless, as mentioned above, the ballots are disclosed). However, even if the adversary's choices show on the receipt as requested, any of them can be the voter's real choice.

Theorem 2.5.1. *Our scheme protects against all of the following.*

- *Loss of votes. The presence of a vote can be verified by the voter if the signatures on the ballots do not collide.*

- *Casting of an incorrect undetected vote by a voting machine. The probability that a local authority will be able to manufacture a fake ballot is equal to the probability to guess an encryption of the fake vote.*

- *Coercion by an adversary that does not have access to the voting machine or the central authority. The receipt bears no information about the actual vote; any voter-controlled parts of the receipt can be selected to match the coercer's request without affecting the voter's choice of candidate.*

- *Tampering with the tally by an adversary that did not have access to the voting machines during the election. The ballots are signed by the voting machines; the probability of malfunction is the probability of collision of the signatures.*

2.5.4 Vulnerabilities

Our scheme is vulnerable in the following respects.

- If the central authority shares the keys with the local authorities before the election, this would allow the local authorities to corrupt ballots without detection.
- If ballots are disclosed, the property of coercion resistance would be lost.
- The central authority can check ballots and post signatures in accordance with the rules, but lie about the tally. Only trusted observers will be able to verify the tally.

2.6 Practical applicability

To cast one vote, the voters will have to submit two long numbers (the ticket identifier and the encrypted index of the real vote) and some short numbers. To avoid this in practice, a ticket could contain subtickets that can be scanned by the machine, containing these long numbers. Alternatively, the vote could be cast electronically (as in the prototype described in the next section).

To simplify the voter's interaction with the voting machine, the permutation of votes could be randomly generated by the machine for the voter. However, allowing this to happen would give the voting machine some control over the contents of the ballot.

Another usability problem is due to the fact that the voter should submit only one key. If the voter changes his mind after submitting this key, he will not be able to change his choice without either leaking information or drawing another ticket.

Finally, in a real election more than one vote usually has to be cast, and more than one ticket will have to be used.

2.7 Prototype

We created a prototype for this scheme. The prototype imitates user interaction with the remote authority and with the local voting machine. It is currently accessible at `althing.`

`dartmouth.edu/cgi-bin/electme2/master.pl`.

The interaction proceeds as follows.

- First, the user opens two browser processes: one simulating the remote authority, another simulating the local voting machine.
- The user requests initialization of the election from the remote authority. The remote authority generates keys and tickets.
- The user selects a random ticket. The remote authority marks the ticket as taken.
- The user submits the ticket ID to the local voting machine. (See Figure 2.3.)
- The local voting machine prompts the user for a permutation of the indices on the ticket corresponding to the placement of the votes ("customization of the ballot"). (See Figure 2.4.)
- The local voting machine then lets the user vote. (See Figure 2.5.)
- The local voting machine generates the receipt consisting of the ticket ID and the indices, and of the signature (the encryption of the hash of the message with the public key of the central authority).
- The local voting machine submits the encryption of the ballot and the signature to the central authority.
- The user can go to the website of the central authority and enter the keys listed under the indices submitted. If the keys have his receipt's vote choices listed under them, he either gets the vote listed back (e.g., Figure 2.6), or knows that his vote did not get submitted correctly (e.g., Figure 2.7).

2.8 Summary of properties of our system

We believe that in real-world situations, our scheme may work to allow voter verification without making the voter susceptible to coercion more than he already is. (For example, in the real-world situation the voter may be coerced not to go to the polling place.) The scheme achieves these results by allowing the use of voter knowledge that cannot be used by a hypothetic coercer.

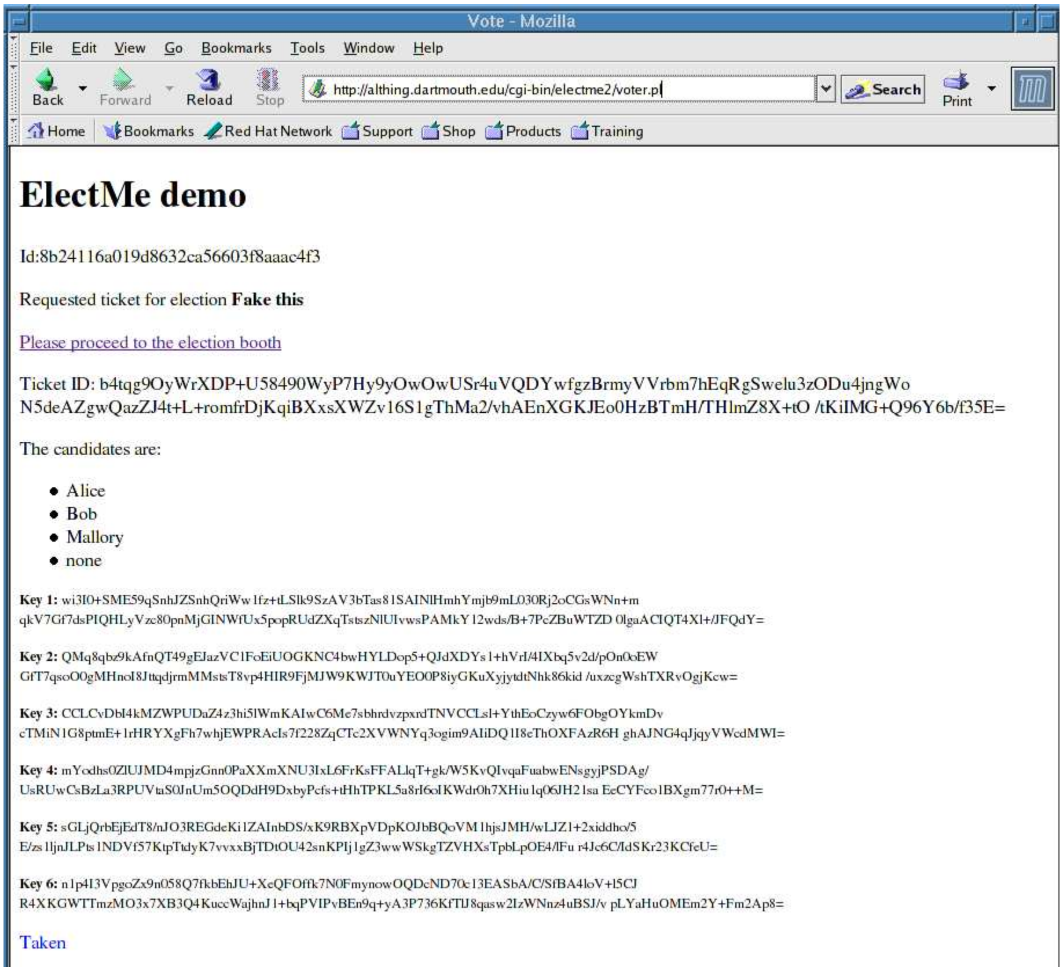
Our scheme takes the possible malfunctioning of or malicious interference with voting machines and loss of votes out of consideration, by replacing it with later verification. Whereas the correctness of VoteHere's scheme depends on the initial step of generation of codebooks and on verification that all voting machines function as intended (which may not be a trivial task), the correctness of our scheme depends only on the secrecy of data distributed before the election. Every voter's vote can be modified only with a small probability, ensuring that if the tampering is detected the votes can be cast again.

Our scheme hands out a readable receipt that allows the voter to see that his vote got cast as intended if the initial conditions of secrecy were met. We believe that this may help with achieving voter trust in the system.

Our scheme is vulnerable if the central authority cheats before the election by cooperating with the local authorities, and if a sufficient audit process is not feasible to verify the tally after the election.

Our scheme is also vulnerable to hypothetical post-election disclosures of ballots. It may be possible to address this issue by providing an extra layer of encryption between the user and the voting machine, at the cost to usability. However, we do not believe that this is likely to be a problem in a situation without a strong adversary.

In future work, we plan to address these shortcomings, as well as carry out pilots with real users, and examine the usability of the various ways of a human user might communicate the keys to the voting machine.


Vote - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://althing.dartmouth.edu/cgi-bin/electme2/voter.pl
Search Print
Home Bookmarks Red Hat Network Support Shop Products Training

ElectMe demo

Id:8b24116a019d8632ca56603f8aaac4f3

Requested ticket for election **Fake this**

Please proceed to the election booth

Ticket ID: b4tqg9OyWrXDP+U58490WyP7Hy9yOwOwUSr4uVQDYwfgzBrmyVVrbm7hEqRgSwelu3zODu4jngWo N5deAZgwQazZJ4t+L+romfrDjKqiBXxsXWZv16S1gThMa2/vhAEnXGKJEo0HzBTmH/THlmZ8X+tO /tKiIMG+Q96Y6b/f35E=

The candidates are:

- Alice
- Bob
- Mallory
- none

Key 1: wi3I0+SME59qSnhJZSnhQriWw1fz+tLSlk9SzAV3bTas81SAINlHmhYmjb9mL030Rj2oCGsWNn+m qkV7Gf7dsPIQHLyVzc80pnMjGINWfUx5popRUdZXqTstszNlUIvwsPAMkY12wds/B+7PcZBuWTZD 0lgaACIQT4Xl+/JFQdY=

Key 2: QMq8qbz9kAfnQT49gEJazVC1FoEiUOGKNC4bwHYLDop5+QJdXDYs1+hVrI/4IXbq5v2d/pOn0oEW GfT7qsoO0gMHnoI8JttqdjrmMMstsT8vp4HIR9FjMJW9KWJT0uYEO0P8iyGKuXyjytdtNhk86kid /uxzcgWshTXRvOgjKcw=

Key 3: CCLCvDbI4kMZWPUDaZ4z3hi5lWmKAIwC6Me7sbhrdvzpxrdTNVCCLsl+YthEoCzyw6FObgOYkmDv cTMiN1G8ptmE+1rHRYXgFh7whjEWPRAcIs7f228ZqCTc2XVWNYq3ogim9AIiDQ1I8eThOXFAzR6H ghAJNG4qJjqyVWcdMWI=

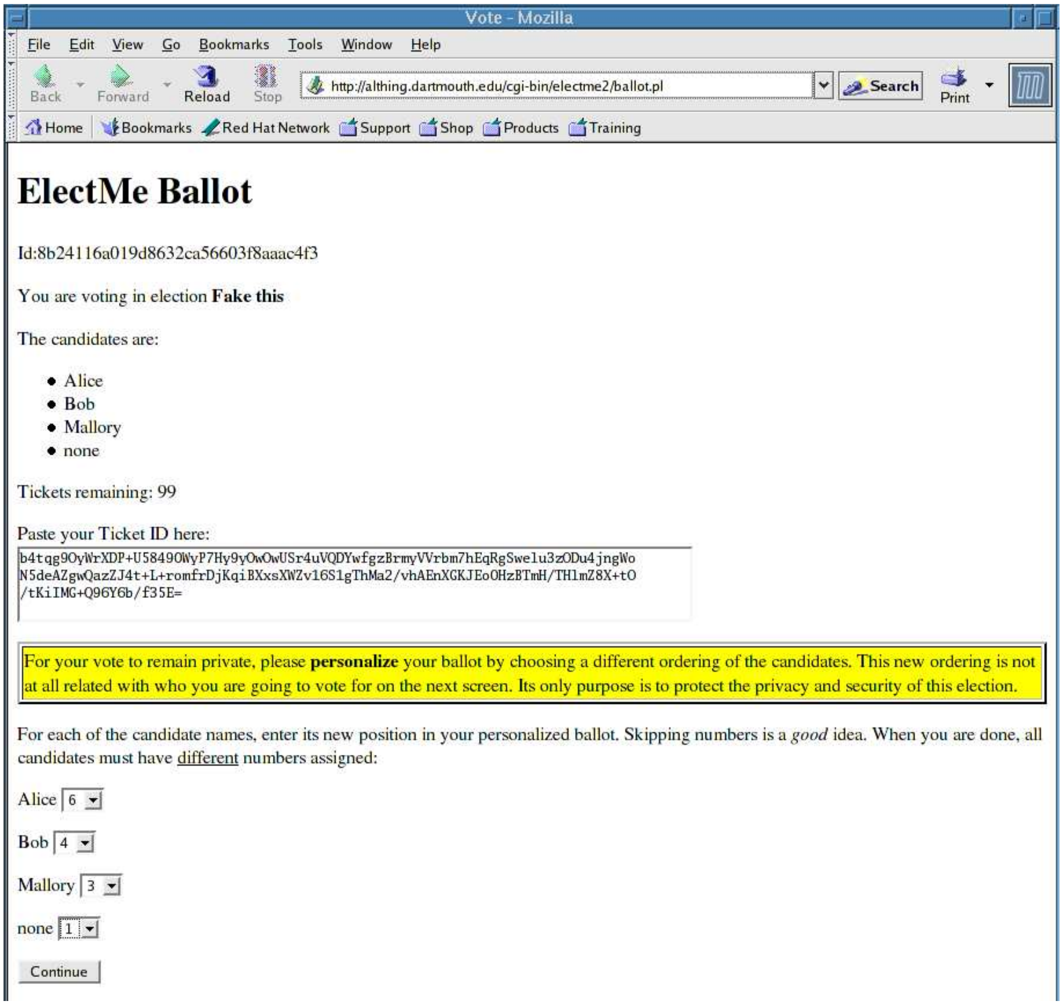
Key 4: mYcdhs0ZlUJMD4mpjzGnn0PaXXmXNU3IxL6FrKsFFALlqT+gk/W5KvQIvqaFuabwENsgyjPSDAg/ UsRUwCsBzLa3RPUVtaS0JnUm5OQDdH9DxbyPcfs+tHhTPKL5a8rI6oIKWdr0h7XHiu1q06JH21sa EeCYFco1BXgm77r0++M=

Key 5: sGLjQrbEjEdT8/nJO3REGdeKi1ZAInbDS/xK9RBXpVDpKOJbBQoVM1hjsJMH/wLJZ1+2xiddho/5 E/zs1ljnJLPts1NDVf57KtpTtdyK7vvxxBjTDtOU42snKPIj1gZ3wwWSkgTZVHXsTpbLpOE4/lFu r4Jc6C/IdSKr23KCfeU=

Key 6: n1p4I3VpgoZx9n058Q7fkbEhJU+XeQFOffk7N0FmynowOQDcND70c13EASbA/C/SfBA4loV+l5CJ R4XKGWTTmzMO3x7XB3Q4KuccWajhnJ1+bqPVIPvBEn9q+yA3P736KfTlJ8qasw2IzWNnz4uBSJ/v pLYaHuOMEm2Y+Fm2Ap8=

Taken

Figure 2.3: The voter submits the ticket to the voting machine


Vote - Mozilla
File Edit View Go Bookmarks Tools Window Help
Back Forward Reload Stop
http://althing.dartmouth.edu/cgi-bin/electme2/ballot.pl
Search
Print
Home Bookmarks Red Hat Network Support Shop Products Training

ElectMe Ballot

Id:8b24116a019d8632ca56603f8aaac4f3

You are voting in election **Fake this**

The candidates are:

- Alice
- Bob
- Mallory
- none

Tickets remaining: 99

Paste your Ticket ID here:

```
b4tqg9OyWrXDP+U5849OWyP7Hy9yOwOwUSr4uVQDYwfgzBrmyVVrbm7hEqRgSwelu3zODu4jngWo
N5deAZgwQazZJ4t+L+romfrDjKqiBXxsXWZv16S1gThMa2/vhAEnXGKJEoOHzBTmH/THlmZ8X+tO
/tKiIMG+Q96Y6b/f35E=
```

For your vote to remain private, please **personalize** your ballot by choosing a different ordering of the candidates. This new ordering is not at all related with who you are going to vote for on the next screen. Its only purpose is to protect the privacy and security of this election.

For each of the candidate names, enter its new position in your personalized ballot. Skipping numbers is a *good* idea. When you are done, all candidates must have different numbers assigned:

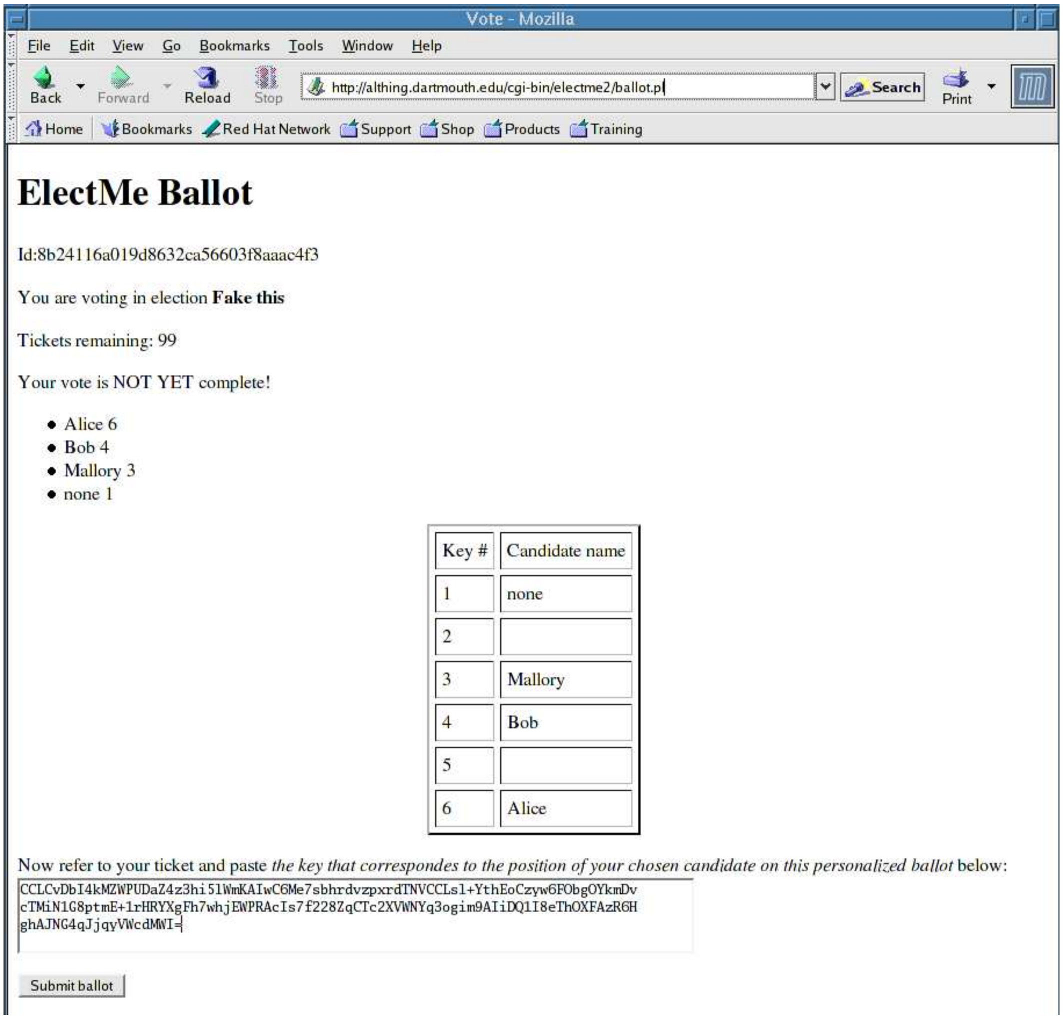
Alice 6

Bob 4

Mallory 3

none 1

Figure 2.4: The voting machine responds with a prompt for the permutation



ElectMe Ballot

Id:8b24116a019d8632ca56603f8aaac4f3

You are voting in election **Fake this**

Tickets remaining: 99

Your vote is NOT YET complete!

- Alice 6
- Bob 4
- Mallory 3
- none 1

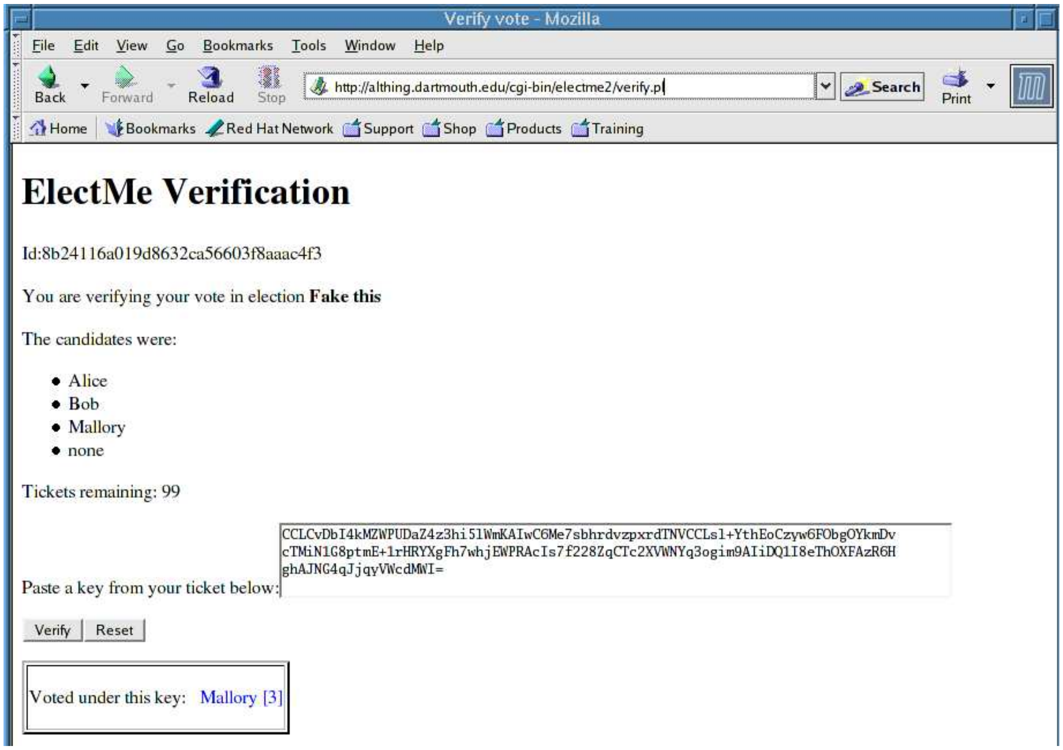
Key #	Candidate name
1	none
2	
3	Mallory
4	Bob
5	
6	Alice

Now refer to your ticket and paste *the key that correspondes to the position of your chosen candidate on this personalized ballot* below:

CCLCvDbI4kMZWPUDaZ4z3hi5lWmKAIwC6Me7sbhrdvzpxrdTNVCCLsl+YthEoCzyw6FObgOYkmDv
cTMiN1G8ptmE+1rHRYXgFh7whjEWPRAcIs7f228ZqCTc2XVWNYq3ogim9AIiDQ1I8eThOXFAzR6H
ghAJNG4qJjqyVWcdMWI=

Submit ballot

Figure 2.5: The voter casts his vote



ElectMe Verification

Id:8b24116a019d8632ca56603f8aaac4f3

You are verifying your vote in election **Fake this**

The candidates were:

- Alice
- Bob
- Mallory
- none

Tickets remaining: 99

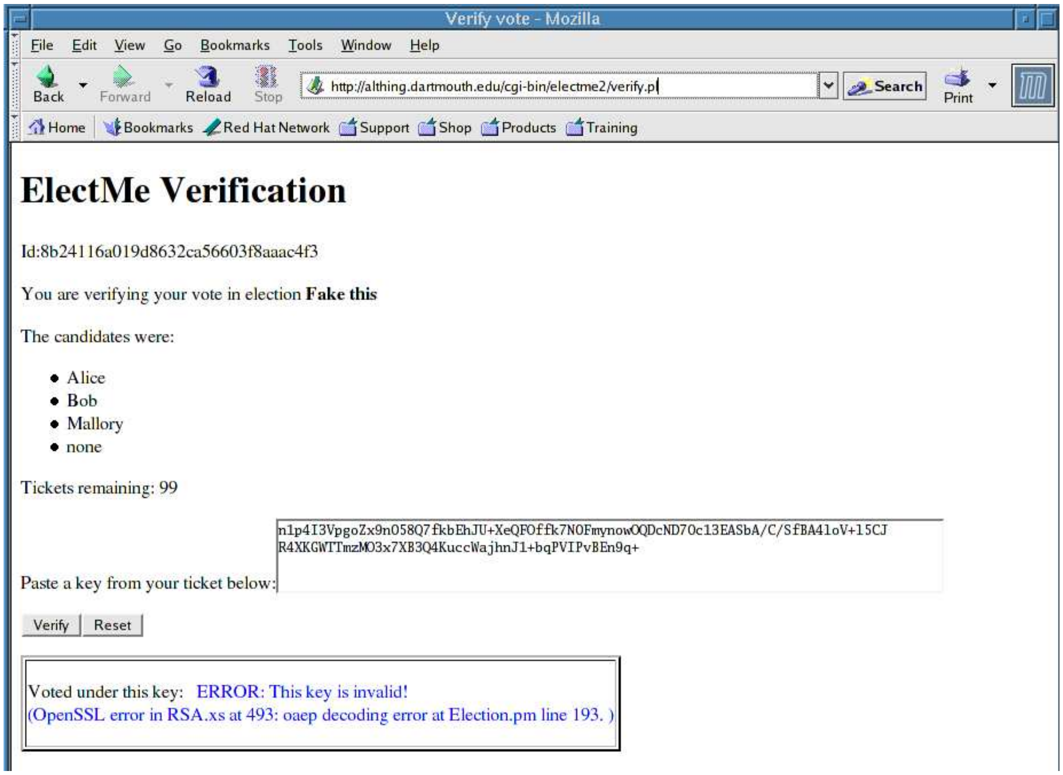
Paste a key from your ticket below:

```
CCLCvDbI4kMZWPUDaZ4z3hi5lWmKAIwC6Me7sbhrdvzpxrdTNVCCLsl+YthEoCzyw6FObgOYkmDv
cTMiN1G8ptmE+1rHRYXgFh7whjEWPRAcIs7f228ZqCTc2XVWNYq3ogim9AIiDQ1I8eThOXFAzR6H
ghAJNG4qJjqyVWcdMWI=
```

Verify Reset

Voted under this key: Mallory [3]

Figure 2.6: The voter checks that a ballot matching his ticket has been cast



http://althing.dartmouth.edu/cgi-bin/electme2/verify.pl

ElectMe Verification

Id:8b24116a019d8632ca56603f8aaac4f3

You are verifying your vote in election **Fake this**

The candidates were:

- Alice
- Bob
- Mallory
- none

Tickets remaining: 99

Paste a key from your ticket below:

```
n1p4I3VpgoZx9n058Q7fkbEhJU+XeQFOffk7N0FmynowOQDcND70c13EASbA/C/SfBA41oV+15CJ
R4XKGWTTmzMO3x7XB3Q4KuccWajhnJ1+bqPVIPvBEn9q+
```

Voted under this key: ERROR: This key is invalid!
(OpenSSL error in RSA.xs at 493: oaep decoding error at Election.pm line 193.)

Figure 2.7: The voter submits an invalid key

Chapter 3

Nearly Private Information Retrieval

A private information retrieval scheme is a protocol whereby a client obtains a record from a database without the database operators learning anything about which record the client requested. The client's request thus remains totally private, uninformative both to the operators of the database and to any external observers. Private information retrieval schemes are well studied in the theoretical computer science literature.

As opposed to the schemes where the client is not concerned about the operators finding out which record he requested, private information schemes require a relatively large number of bits to be transmitted in order to hide the client's real intent. This limits the practical applicability of such schemes. Consequently, the main question answered in private information retrieval literature is the question of the amount of communication required in such protocols.

In the real world situations, the requirement of total privacy is frequently excessive. If only a small number of requests to a large database is made, it may be acceptable to allow leaking some information about the record requested. In extreme cases, the user sometimes may not even be so much concerned with privacy as with plausible deniability, with the impossibility to prove that a certain request was made.

In this chapter we study a generalization of the private information retrieval concept to nearly private information retrieval: to protocols that allow a small amount of information about the client's intent to be leaked.

Despite having relaxed the privacy requirement, we were able to prove three fairly strong lower bounds on amounts of communication in nearly private information retrieval schemes, for various parameter settings. These bounds extend previously known lower bounds in the traditional setting of perfect privacy and, in one case, improve upon the previous best result that handled imperfect privacy.

3.1 Private information retrieval

Private information retrieval (PIR) schemes have been a substantial focus of theoretical research in computer science, beginning with the highly influential work of Chor, Goldreich, Kushilevitz and Sudan [CGKS98]. In that paper, as in most subsequent work, a PIR scheme has been used to mean a communication protocol that specifies an interaction between a *client* or *user* and one or more *servers*. The user wishes to obtain a record from the database without the servers learning anything about which record the user seeks.

This problem stems from the same source as the problem of anonymous communication over the internet discussed in Chapter 1. A user may have many legitimate reasons to conceal the details of his request for information. Examples of searches for sensitive information include medical data, lookups in patent databases, requests of stock market data, and various searches necessitated by intelligence gathering.

It is worth noting that the practical applicability of PIR schemes has been discussed in literature. In particular, in [SC07], Radu Sion et al argue that any non-trivial single-server PIR protocol on real hardware is less efficient than transmitting the entire database.

A clean and concrete version of this problem, as proposed by Chor et al., is as follows: the database y is a string of n bits. The client has an index $j \in \{1, 2, \ldots, n\}$ and she wishes to obtain the jth bit of y, without the servers learning any information about j. As shown by Chor et al., this strong privacy requirement means that if there is only one server that holds the database, the trivial protocol in which the client simply downloads the entire database is optimal in terms of the number of bits communicated. However, as shown in the same paper, if one allows the database to be replicated and copies held by two or more servers that do not talk to each other, the problem can be solved using sublinear communication.

Almost all past work on PIR schemes has required that the servers learn *zero* information about the client's index j. Here, we ask the question: what happens if we allow the protocol to leak a small amount of information about j? To the best of our knowledge, the only other work to have considered this question is that of Goldreich, Karloff, Schulman and Trevisan [GKST02]. It is *a priori* conceivable that relaxing the privacy requirement might decrease the communication required in PIR protocols. However, in this work, we prove three lower bounds that show that previously known lower bounds for traditional (perfect privacy) PIR protocols extend to this relaxed setting, up to constant factor losses. One of our bounds improves an earlier result of Goldreich et al. from the aforementioned paper. We also show that another of our bounds is essentially optimal by exhibiting an appropriate upper bound.

To explain our results in detail and compare them to previously known results, we begin with some necessarily definitions.

3.1.1 Preliminaries

Notation

We begin by introducing some notation.

We denote $[n]$ to be the set $\{1, 2, \ldots, n\}$.

For random variables X and Y that take values in the same set S, we write $X \approx_\delta Y$ to mean that the L_1 distance between the distributions of X and Y is at most δ. To be precise,

$$\sum_{a \in S} |\Pr[X = a] - \Pr[Y = a]| \ \leq \ \delta \, .$$

Definition Let s, ℓ_q, ℓ_a be positive integers and ε, δ be reals in $[0, 1]$. An 1-round s-server $(\ell_q, \ell_a; \varepsilon, \delta)$-PIR protocol $\mathcal{P}$ is a communication protocol between a single client, which holds an index $j \in [n]$ and s servers, each of which holds a string $y \in \{0, 1\}^n$. Formally, $\mathcal{P}$

is specified by the following set of functions.

- Query functions $Q_i : [n] \times \{0,1\}^\rho \longrightarrow \{0,1\}^{\ell_q}$, from the user to servers $S_1, S_2, \ldots, S_k$ where the input of Q_j is the index i and a random input of length m, and the output of Q_j is the query string that the user should send to server S_j.
- Answer functions $\mathrm{Ans}_i : \{0,1\}^n \times \{0,1\}^{\ell_q} \longrightarrow \{0,1\}^{\ell_a}$ that compute the string to be returned to the user by server j from the database and from the user's query.
- Recovery function $\mathrm{Rec} : [n] \times \{0,1\}^\rho \times [\{0,1\}^{\ell_a}]^k \longrightarrow \{0,1\}$ that computes the ith bit of the database from i, the user's randomness, and the answers of the servers to the user's queries.

For compactness of notation, we shall drop the subscript on Q_i and Ans_i altogether when $s = 1$. The protocol operates as follows: the client generates a random string R distributed uniformly in $\{0,1\}^\rho$ and, for each $i \in [n]$, sends a *query* string $Q_i(j, R)$ to server i. Upon receiving a query string z, server i sends an *answer* string $\mathrm{Ans}_i(y, z)$ to the client. The client then outputs a *recovered bit*

$$\mathrm{Out}(j, y, R) \;:=\; \mathrm{Rec}(j, R, \mathrm{Ans}_1(y, Q_1(j, R)), \ldots, \mathrm{Ans}_s(y, Q_s(j, R)))$$

which is the client's guess at the value of y_j. The protocol must satisfy the following two conditions.

Correctness: $\forall\, j \in [n], y \in \{0,1\}^n : \;\Pr_R[\mathrm{Out}(j, y, R) = y_j] \geq 1 - \varepsilon$.

Privacy: $\forall\, i \in [s],\, j, k \in [n] : \;Q_i(j_1, R) \approx_\delta Q_i(j_2, R)$.

The parameter q is called the *query length*, a the *answer length*, ε the *recovery error*, and δ the *privacy parameter* of the protocol $\mathcal{P}$. The communication cost of $\mathcal{P}$ is $\mathrm{cost}(\mathcal{P}) = s(\ell_q + \ell_a)$, the total number of bits communicated. Since we only deal with constant values of s, there is no asymptotic loss in requiring that all servers receive queries of the same length and return answers of the same length. The goal in designing PIR protocols is to simultaneously reduce ε, δ, and $\mathrm{cost}(\mathcal{P})$.

When $\varepsilon = 0$, the protocol is said to have *perfect recovery* and when $\delta = 0$, it is said to have *perfect privacy*. The bulk of theoretical research on PIR has focused on the case $\varepsilon = \delta = 0$. The work of Goldreich et al. [GKST02] and that of Kerenidis and de Wolf [KdW03] did consider the $\varepsilon > 0$ case. However, relatively little attention has been paid to the $\delta > 0$ case, except for one result of Goldreich et al. mentioned below.

When $s = 1$, we shall drop the first arguments of the functions Q and Ans. Thus, we will have $Q : [n] \times \{0, 1\}^{\rho} \to \{0, 1\}^{q}$ and $\text{Ans} : \{0, 1\}^{n} \times \{0, 1\}^{q} \to \{0, 1\}^{a}$.

Quantum systems

An in-depth explanation of quantum computation can be found, for example, in [NC00]. We present the basics here, in order to be able to follow and extend the quantum argument of Kerenidis and de Wolf in [KdW03].

A *qubit* $|\phi\rangle$ is a vector of unit length in a 2-dimensional complex vector space with a standard basis, the vectors of which are denoted by $|0\rangle$ and $|1\rangle$. Namely,

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle,$$

where α and β are called *amplitudes*, and $|\alpha|^2 + |\beta|^2 = 1$.

The qubit $|\phi\rangle$ can be *measured* in an orthonormal basis. If we measure it in the standard basis, the result of this measurement will be either $|0\rangle$ with probability $|\alpha|^2$ or $|1\rangle$ with probability $|\beta|^2$.

We will consider also *m-qubit states*, which are unit vectors in a tensor product of m 2-dimensional vector spaces such as described above. Such vectors can be denoted by $|b_1\rangle \otimes \ldots \otimes |b_m\rangle$, $|b_1\rangle \ldots |b_m\rangle$, or $|b_1 \ldots b_m\rangle$, where $b_1, \ldots, b_m \in \{0, 1\}$. The inner product of two such states $|\phi\rangle$ and $|\psi\rangle$ is denoted by $\langle\phi|\psi\rangle$ and defined in the ordinary Euclidean sense. Two states are said to be orthogonal if their inner product is 0. An m-qubit state $|\phi\rangle$ can also be measured in an orthonormal basis of $|\psi_1\rangle, \ldots, |\psi_n\rangle$, giving the outcome of $|\psi_i\rangle$ with the probability $\langle\psi_i|\phi\rangle^2$. If the outcome is $|\psi_i\rangle$, the state will then collapse to the state $|\psi_i\rangle$.

Kerenidis and de Wolf consider *quantum queries*. A quantum query to a string x can be defined as a transformation

$$|c\rangle|i\rangle \mapsto (-1)^{c\cdot x_i}|c\rangle|i\rangle,$$

where c is called a *control bit*, because it specifies whether or not the phase $(-1)^{x_i}$ is added to the vector.

Locally decodable codes

Definition We call an encoding $C : \{0,1\}^n \longrightarrow \{0,1\}^m$ a (q, δ, ϵ)-*locally decodable code* (or *LDC*) if there exists a randomized decoding algorithm that queries the codeword non-adaptively at most q times and decodes each separate bit correctly with probability $\geq 1-\epsilon$ if the encoding is corrupted in no more than δm bits.

Such an encoding is called a (q, δ, ϵ)-*locally quantum-decodable code (LQDC)* if the algorithm is allowed to be a quantum computer and to make queries in superposition.

3.1.2 Previous work

In the 2-server case, it is known that $O(n^{1/3})$ communication can be achieved, even with $\varepsilon = \delta = 0$, via a number of different schemes; see, e.g., Chor et al. [CGKS98], Beimel, Ishai and Malkin [BIM00], and Woodruff and Yekhanin [WY05]. No strong general lower bound is known that comes close to matching this upper bound. However, a recent result of Razborov and Yekhanin [RY06] provides an $\Omega(n^{1/3})$ bound for protocols whose computation is restricted in a certain way (bilinear group based protocols, which include all known 2-server protocols). With arbitrary computations allowed, there *are* strong lower bounds known provided the answer length ℓ_a is short. The cleanest of these results are for the $\ell_a = 1$ case. In this case, Kerenidis and de Wolf [KdW03] prove a lower bound of $(1 - H(11/14 - 4\varepsilon/7))\, n - 2$ when $\delta = 0$. Beigel, Fortnow and Gasarch [BFG06] prove a tight $n - 2$ lower bound when $\varepsilon = \delta = 0$.

A lower bound handling positive ε and δ was proven for 2-server case by Goldreich et al. [GKST02]. Their bound, for $\ell_a = 1$, is $(1 - 2\varepsilon - \delta)\, n/24 - 4$. (Note that our use of ε and δ is different from theirs; we have translated their bound into our notation.)

It is worth noting that the issue of lower bounds for PIR schemes with 3 or more servers has recently been largely settled, in a most dramatic way, by Yekhanin [Yek07]. Yekhanin proved that if there are an infinite number of Mersenne primes, then for infinitely many n there is an $O(n^{1/\log\log n})$ 3-server PIR protocol.

Computationally bounded private information retrieval

The above results refer to traditional PIR that assumes no bounds on computational power of the database servers. PIR schemes with databases that have computational limits have also been considered in the literature. Number theoretic conjectures and one-way functions have been used to achieve PIR schemes with communication that, in some cases, beat the known upper bounds, and in other beat the known lower bounds. In particular, Kushilevitz and Ostrovsky used the hardness of the Quadratic Residue Problem in [KO97] and one-way trapdoor permutations in [KO00], whereas Stern [Ste98] and Mann [Man98] used homomorphic encryption to generate 1-database PIR schemes with $O(n^{\epsilon})$ communication, where ϵ can be arbitrarily small. In [CG97], Chor and Gilboa used one-way functions to construct 2-database schemes with $O(n^{\epsilon})$ communication. This thesis does not study computationally limited PIR.

3.1.3 Our results

We prove three lower bounds that allow $\delta > 0$. Let P be a 1-server $(\ell_q, \ell_a; \varepsilon, \delta)$-PIR protocol. With the privacy requirement relaxed, even the 1-server case becomes nontrivial and it is not *a priori* clear that sublinear communication PIR is not possible. However, we show that for $\varepsilon = 0$, we must have

$$\mathrm{cost}(P) \geq (1 - \delta/2)\, n = \Omega(n).$$

We also show, via an upper bound, that this dependence to δ is essentially tight, up to terms quadratic in δ.

We also consider the more general case when both ε and δ can be nonzero. In this case, we show that $\mathrm{cost}(P) \geq (1 - H(\varepsilon + \delta/2))\, n$ for sufficiently small ε and δ. Here H is the binary entropy function given by $H(x) := -x \lg x - (1-x)\lg(1-x)$; "lg" denotes logarithm to the base 2.

Finally, we consider 2-server nearly private information retrieval schemes. Here, we prove a lower bound of $(1 - H(3/4 + 2\delta/3 - \sqrt{2\delta} - \varepsilon))\, n - 2$ when $\ell_a = 1$, for sufficiently small positive ε and δ. To see that our bound is an improvement over the bound of Goldreich and co, consider the limiting case $\varepsilon \to 0$, $\delta \to 0$: our lower bound then approaches $0.19n - 2$, whereas the bound of [GKST02] approaches $0.04n - 4$.

3.2 Upper bounds

Here we show simple improvements to the known upper bounds on the communication cost of PIR schemes by allowing imperfect privacy. As we shall see later, the 1-server upper bound we obtain below is essentially optimal in the perfect recovery case.

In the perfectly private case, the obvious upper bound is n bits, achieved, for example, when the user sends no bits at all and downloads the whole database to guarantee his privacy. This bound is equal to the lower bound proven in the original PIR paper [CGKS95]. Below we show how to beat this bound in the nearly private case.

Theorem 3.2.1. *For any $\delta > 0$, there is a PIR protocol with perfect recovery, privacy parameter δ and communication cost at most $\lceil \lg n \rceil + \lceil (1 - \delta/(2+\delta))\, n \rceil = (1 - \delta/2 + O(\delta^2))\, n + O(\log n)$.*

Proof. Let $\delta' = \delta/(2+\delta)$. For each integer $j \in [n]$, define the sets

$$\begin{aligned} S_j &:= \{k \in [n] : 0 \leq (k - j) \bmod n \leq (1 - \delta')n\}, \\ T_j &:= \{k \in [n] : 0 \leq (j - k) \bmod n \leq (1 - \delta')n\}. \end{aligned}$$

Design the function Q so that, when R is a uniform random string, $Q(j, R)$ is uniformly distributed on S_j. For $k \in [n]$ and $y \in \{0, 1\}^n$, let $\mathrm{Ans}(y, k)$ return the concatenation, in some canonical order, of all y_j such that $j \in T_k$. It is easy to see that $k \in S_j \Leftrightarrow j \in T_k$; therefore $\mathrm{Ans}(y, Q(j, R))$ is guaranteed to contain the desired bit y_j and we can design Rec so as to recover y_j from $Q(j, R)$ and $\mathrm{Ans}(y, Q(j, R))$. Clearly, the PIR protocol given by $(Q, \mathrm{Ans}, \mathrm{Rec})$ has perfect recovery and communication cost at most $\lceil \lg n \rceil + |T_k| \leq \lceil \lg n \rceil + \lceil (1 - \delta')n \rceil$.

For all $j \in [n]$, we have $|S_j| \geq (1 - \delta')n$ and for $i \neq j$, we have $|S_i \setminus S_j| + |S_j \setminus S_i| \leq 2 \cdot |[n] \setminus S_j| \leq 2\delta' n$. Therefore, we can bound the protocol's privacy parameter as follows:

$$Q(i, R) \approx_{\delta''} Q(j, R), \quad \text{where } \delta'' \leq \frac{2\delta' n}{(1 - \delta')n} = \delta . \qquad \square$$

A 2-server upper bound. In a similar manner to the 1-server case, it is possible to add a δ-dependent coefficient to the $O(n^{1/3})$ upper bound for 2-server PIR. The idea is to suitably modify the covering codes scheme of Chor et al. [CGKS98].

In [CGKS95], the covering codes scheme for 2 servers is described in the following way.

Associate the database with a 3-dimensional cube so that each side of the cube is of length $\sqrt[3]{n}$, and the ith element is at i_1, i_2, i_3. Consider, for the moment, 2^3 servers corresponding to all vertices of the cube and denote them by $\mathcal{S}_{\sigma_1\sigma_2\sigma_3}$. Randomly generate $S_1^0, S_2^0, S_3^0 \subset [\sqrt[3]{n}]$ and define $S_1^1 = S_1^0 \oplus i_1$, $S_2^1 = S_2^0 \oplus i_2$, $S_3^1 = S_3^0 \oplus i_3$. For each $\sigma_1, \sigma_2, \sigma_3$ send $S_1^{\sigma_1}, S_2^{\sigma_2}, S_3^{\sigma_3}$ to $\mathcal{S}_{\sigma_1\sigma_2\sigma_3}$, and have the server respond with

$$\bigoplus_{j_1, j_2, j_3 \in S_1^{\sigma_1} \times S_2^{\sigma_2} \times S_3^{\sigma_3}} x_{j_1, j_2, j_3} .$$

The user can then reconstruct the x_{i_1, i_2, i_3} by taking an exclusive-or of all responses. Chor et al. use S_{000} and S_{111} to emulate servers $S_{001}, S_{010}, S_{100}$ and $S_{011}, S_{110}, S_{101}$ respectively in the following way. To emulate S_{100}, for example, the server S_{000} sends all the $\sqrt[3]{n}$ responses corresponding to all the possibilities for i_1; the other servers would be emulated similarly.

Thus, each of S_{000} and S_{111} will send 1 bit for itself and $\sqrt[3]{n}$ bits for each of the 3 emulated servers, for the total of $2 + 6\sqrt[3]{n}$ bits of communication). The user had to start by sending S_1^0, S_2^0, S_3^0 to S_{000} and S_1^1, S_2^1, S_3^1 to S_{111} for the total of $6\sqrt[3]{n}$ bits; thus the total is $2+12\sqrt[3]{n}$.

Extending our scheme from the previous section, all we need to do is select a subcube containing $(1-\delta')n$ indices and guaranteed to contain the index that we are interested in. We select it by having the user send $\log n$ bits to the 2 servers and then proceed with the previous scheme, for the total of $2 + 2\log n + 12\sqrt[3]{(1-\delta')n}$ communication.

3.3 1-Server lower bounds

3.3.1 Perfect privacy and recovery

Chor et al. [CGKS98] prove that, in the 1-server case with perfect privacy, n bits must be exchanged. Their argument goes as follows. A communication C (the string of exchanged bits) is said to be *possible* for (y, j) if there is a positive probability for C to happen when the database is y, and the user tries to obtain the jth bit. C is said to be *possible* for j if it is possible for some pair (y, i). Let us fix a j and assume that the number of possible communications for j is less than 2^n. Then there exist different databases y, y' and C such that C is possible for both (y, j) and (y', j). But by the privacy requirement, for every $k \in [n]$, C must also be possible for (y, k) and (y', k), since the queries are distributed equally, and the responses are determined by the queries. Pick an index j such that $y_j \neq y'_j$. We know that C is possible for both (y, j) and (y', j), but C determines the output of the protocol, thus the protocol must yield the same bit, and we get a contradiction.

This argument fails in the nearly private case, since there is no requirement that the same communication be possible for all indices if it is possible for one. However, we can still obtain strong lower bounds, as we will show below.

3.3.2 Plausible deniability

As mentioned before, the requirement of perfect privacy is very strong and generally unnecessary in the real world, where people may be concerned more with plausible deniability than with perfect privacy.

More specifically, instead of saying that for all indices the distributions of possible communications must be the same, in the real world it may be sufficient to settle for the same set of communications being possible for every index.

The argument of Chor et al. [CGKS98], as described in the previous section, goes through for this case without any changes, since it only requires the same set of communications being possible for every index and makes no use of the requirement that these communications be equally distributed for every index. It turns out that *n* bits of communication is required merely to provide plausible deniability.

3.3.3 Nearly private schemes with perfect recovery

Theorem 3.3.1. *Let $\mathcal{P}$ be a* 1*-server* $(\ell_q, \ell_a; 0, \delta)$*-PIR protocol, where* $\delta > 0$*. Then* $\ell_a \geq (1 - \delta/2)n$*. In particular,* $\mathrm{cost}(\mathcal{P}) \geq (1 - \delta/2)n$*.*

Proof. For an index $j \in [n]$ and a query $z \in \{0, 1\}^q$, let $p_{jz} = \mathrm{Pr}_R[Q(j, R) = z]$. Let $J_z = \{j : p_{jz} > 0\}$. If $p_{jz} > 0$, y_j must be recoverable from the response to query z, hence J_z is greater or equal the number of database elements that should be recoverable from the response to query z. It is easy to verify that

$$|J_z|p_{1z} \;\geq\; \sum_{j=1}^{n} \min\{p_{1z}, p_{jz}\} \;=\; \sum_{j=1}^{n} \left(\frac{p_{1z} + p_{jz}}{2} - \frac{|p_{1z} - p_{jz}|}{2} \right) .$$

This implies that

$$\sum_{z\in\{0,1\}^q} |J_z|p_{1z} \;\geq\; \sum_{j=1}^{n} \sum_{z\in\{0,1\}^q} \left(\frac{p_{1z} + p_{jz}}{2} - \frac{|p_{1z} - p_{jz}|}{2} \right) \;\geq\; \sum_{j=1}^{n} \left(\frac{1+1}{2} - \frac{\delta}{2} \right) \;=\; (1{-}\delta/2)n\,,$$

where the final inequality follows from the privacy guarantee of $\mathcal{P}$. Since $\sum_{z\in\{0,1\}^q} p_{1z} = 1$, this means that there exists a $z \in \{0,1\}^q$ such that $|J_z| \geq (1-\delta/2)n$. Fix such a z.

Suppose $\ell_a < |J_z|$. Let $Y := \{y \in \{0,1\}^n : y_j = 0 \text{ for } j \notin J_z\}$. Then $|Y| = 2^{|J_z|}$. The string $\mathrm{Ans}(y,z)$ has length ℓ_a, therefore it lies in a set of size $2^{\ell_a} < 2^{|J_z|}$. By the pigeonhole principle, there exist distinct strings $y, y' \in Y$ such that $\mathrm{Ans}(y,z) = \mathrm{Ans}(y',z)$. Let j be an index such that $y_j \neq y'_j$. Then $j \in J_z$. Therefore, $p_{jz} > 0$, i.e., there exists an R such that $Q(j,R) = z$. Since $\mathcal{P}$ has perfect recovery, for this R we must have

$$y_j \;=\; \mathrm{Rec}(j, R, \mathrm{Ans}(y,z)) \;=\; \mathrm{Rec}(j, R, \mathrm{Ans}(y',z)) \;=\; y'_j\,,$$

which is a contradiction. This proves that $\ell_a \geq |J_z| \geq (1-\delta/2)n$. □

3.3.4 Nearly private schemes with imperfect recovery

We now turn to the imperfect recovery case. We prove our lower bound for this case by a reduction from a communication problem with a well known lower bound. Later, we use a much more sophisticated version of the same idea for a 2-server lower bound.

The problem $\textsc{index}_n$ is a communication problem involving two players: Alice, who holds an n-bit string $x = x_1 x_2 \ldots x_n$ (with each $x_j \in \{0,1\}$) and Bob, who holds an index $j \in [n]$. A one-way communication protocol for this problem operates as follows: Alice sends Bob a message based on x after which Bob outputs his guess at the bit x_j. Both players may use a public random string in making their decisions, i.e., the protocol is allowed to be public coin. Ablayev [Abl96] proved the following sharp lower bound on the communication cost of such a protocol.

Fact 3.3.2. *Any public coin one-way communication protocol for* $\textsc{index}_n$ *with error at most* ε *must communicate at least* $(1 - H(\varepsilon))\,n$ *bits.*

Theorem 3.3.3. *Let* ε *and* δ *be positive reals with* $\varepsilon + \delta/2 < 1/2$. *Then any* 1*-server* $(\ell_q, \ell_a; \varepsilon, \delta)$*-PIR protocol has* $\ell_a \geq (1 - H(\varepsilon + \delta/2))\,n$. *In particular, the communication cost of such a protocol is at least* $(1 - H(\varepsilon + \delta/2))\,n$.

Proof. Suppose P is a 1-server $(\ell_q, \ell_a; \varepsilon, \delta)$-PIR protocol that uses ρ bits of randomness.

Let $\mathcal{D}_{jz}$ denote the conditional distribution of R given that $Q(j, R) = z$ and let Gen : $[n] \times \{0,1\}^{\ell_q} \times \{0,1\}^{\rho'} \to \{0,1\}^{\rho}$ be such that $\mathrm{Gen}(j, z, R')$ is distributed according to $\mathcal{D}_{jz}$ when R' is distributed uniformly in $\{0,1\}^{\rho'}$. Further, define $f : [n] \times \{0,1\}^n \times \{0,1\}^{\ell_q} \times \{0,1\}^{\rho'} \to \{0,1\}$ as follows.

$$f(j, y, z, r') := \begin{cases} 0\,, & \text{if } \mathrm{Rec}(j, \mathrm{Gen}(j, z, r'), \mathrm{Ans}(y, z)) = y_j\,, \\ 1\,, & \text{otherwise}\,. \end{cases}$$

The correctness condition for P implies

$$\begin{aligned} \mathrm{E}_{R,R'}[f(j, y, Q(j, R), R')] &= \Pr_{R,R'}\left[\mathrm{Rec}(j, \mathrm{Gen}(j, Q(j, R), R'), \mathrm{Ans}(y, Q(j, R))) \neq y_j\right] \\ &= \Pr_{R}\left[\mathrm{Rec}(j, R, \mathrm{Ans}(y, Q(j, R))) \neq y_j\right] \\ &\leq \varepsilon\,. \end{aligned}$$

Now, using the privacy condition $Q(j, R) \approx_\delta Q(1, R)$ and the fact that R and R' are independent, we have

$$\mathrm{E}_{R,R'}[f(j, y, Q(1, R), R')] \leq \varepsilon + \frac{\delta}{2}\,.$$

In other words, the following is a public coin one-way communication protocol for the problem INDEX$_n$, with error at most $\varepsilon + \delta/2$. Alice and Bob share a pair of random strings (R, R') distributed uniformly in $\{0,1\}^{\rho} \times \{0,1\}^{\rho'}$. Alice, upon receiving y, sends Bob the message $\mu := \mathrm{Ans}(y, Q(1, R))$. Bob, upon receiving j and μ, outputs $\mathrm{Rec}(j, \mathrm{Gen}(j, Q(1, R), R'), \mu)$ as his guess at y_j. Clearly, this protocol has cost at most ℓ_a. By Fact 3.3.2, we have $\ell_a \geq (1 - H(\varepsilon + \delta/2))\, n$, which completes the proof. □

3.4 2-Server lower bounds

We now turn to the case of 2-server PIR protocols. As mentioned earlier, significantly less is known about lower bounds for such protocols. In particular, the only strong lower bounds known for protocols that may make arbitrary computations are when the answer size is restricted to be quite small. In particular, there are strong results known for the case of one-bit answers.

3.4.1 Prior work on perfectly private schemes

According to Gasarch's survey [Gas04], the following lower bounds exist for the 2-server case.

1. If only linear answers of length a are allowed, then the user must send a query of length $\Omega(\frac{n}{2^n})$. This result is due to Goldreich et al. ([GKST02]).

2. If only linear queries are allowed, and each database returns 1 bit, then the database must get a query of length at least $n-1$. This is due to Chor et al. ([CGKS95]).

3. A quantum argument by Kerenidis and de Wolf in [KdW03] shows that if the user uses only a of the bits sent back, then he must send a query of at least $\Omega(n/2^{6a})$. In the case of 1-bit answers and recovery probability of $1/2+\delta$ the length of the queries has to be at least $(1-H(1/2+4\delta/7))n-2$.

4. In [BFG06] Beigel et al. have proven that in the case of answers of length 1 the queries have to be of length at least $n-2$.

5. The last bound has been published in [RY06] after Gasarch's survey and is due to Razborov and Yekhanin. This bound is $\Omega(n^{1/e})$ in case of bilinear schemes. This bound shows that the $O(n^{1/3})$ upper bound of the currently known protocols is optimal at least for the bilinear case. The paper claims also that all existing 2-server PIR schemes are bilinear-based; the authors prove this claim for some of the schemes.

We shall show an asymptotically optimal lower bound extending the Kerenidis and de Wolf [KdW03] result for the case of one-bit answers to the case of imperfect privacy.

Our proof uses a *quantum computation* framework first used by Kerenidis and de Wolf [KdW03]. Below, we quickly review their framework and argument and then show how to extend it to allow imperfect privacy.

3.4.2 The perfectly private 1-bit case

Kerenidis and de Wolf prove a number of communication lower bounds for 2-server PIR schemes. However, their arguments only handle the perfect privacy case, although they do handle imperfect recovery. Their arguments are cast in a quantum communication framework, the key observation of which can be expressed thus: "a single quantum query can simulate two classical queries." More precisely, the following theorem holds.

Theorem 3.4.1. *(Kerenidis & de Wolf) Given a classical* 2*-server PIR scheme with* t*-bit queries,* 1*-bit answers, and recovery probability* $1/2 + \delta$*, it is possible to produce a quantum* 1*-server PIR scheme with* $(t + 2)$*-qubit queries,* $(t + 2)$*-qubit answers, and recovery probability* $1/2 + 4\delta/7$*.*

Using this observation, Kerenidis and de Wolf build from a 2-server PIR scheme a 1-server quantum PIR scheme, generate a random access code, and then prove lower bounds on the communication in a way analogous to our 1-server lower bounds. In particular, the appropriate quantum analog of Ablayev's lower bound (Fact 3.3.2) turns out to be the following lower bound for quantum random access codes, due to Nayak [Nay99].

Theorem 3.4.2. *(Nayak) For an encoding* $x \mapsto \rho_x$ *of* n*-bit strings into* m*-qubit states with recovery probability of at least* p*,* m *has to be* $\geq (1 - H(p))n$*.*

Kerenidis and de Wolf use the following theorem to reduce locally decodable codes to locally quantum decodable codes.

Theorem 3.4.3. *(Kerenidis & de Wolf) A* $(2, \delta, \epsilon)$*-LDC is a* $(1, \delta, 4\epsilon/7)$*-LQDC.*

Finally, Kerenidis and de Wolf apply Theorem 3.4.2 to get the lower bound.

We now outline Kerenidis and de Wolf's argument, using our own terminology. We find it convenient to remove the intermediate steps of a quantum PIR scheme and a quantum random access code; instead, we show that a 2-server PIR scheme with sufficiently good parameters implies a one-way *quantum* communication protocol for INDEX_n. The aforementioned result of Nayak [Nay99] can be restated thus.

Fact 3.4.4. *A one-way quantum communication protocol for* INDEX_n *with error probability* ε *must communicate at least* $(1 - H(\varepsilon))\,n$ *qubits.*

This gives us the desired PIR lower bound.

The following theorem is the 1-bit result of Kerenidis and de Wolf as restated in our notation.

Theorem 3.4.5. *Any* 2*-server* $(\ell_q, 1; \varepsilon, 0)$*-PIR protocol has* $\mathrm{cost}(P) \geq \ell_q \geq (1-H(11/14-4\varepsilon/7))\,n - 2$.

Proof. Suppose P is a 2-server $(\ell_q, 1; \varepsilon, \delta)$-PIR protocol, given by $(Q, \mathrm{Ans}, \mathrm{Rec})$, that uses ρ bits of randomness. We associate with P a certain collection $\{|\phi_{jy}\rangle\}$ of $(\rho+4+\ell_a)$-qubit quantum states. To define these, we use the basis states $\{|r, i, i, z\rangle : r \in \{0,1\}^\rho, i \in \{0,1,2\}, z \in \{0,1\}^{\ell_q}\}$. We set $c := 1/\sqrt{3 \cdot 2^\rho}$ and, for notational convenience, we define $Q_0(j,r) = 0^{\ell_q}$ and $\mathrm{Ans}_0(y,z) = 0$ for all $j \in [n], r \in \{0,1\}^\rho, y \in \{0,1\}^n$ and $z \in \{0,1\}^{\ell_q}$. Also, for $(i,j,z) \in \{0,1,2\} \times [n] \times \{0,1\}^{\ell_q}$, we define the set $S_{ijz} := \{r \in \{0,1\}^\rho : Q_i(j,r) = z\}$. Finally, we define $|\phi_{jy}\rangle$ as follows:

$$|\phi_{jy}\rangle \ := \sum_{r \in \{0,1\}^\rho} c\,|r\rangle \left(|0,0,0^{\ell_q}\rangle + (-1)^{\mathrm{Ans}_1(y,Q_1(j,r))}|1,1,Q_1(j,r)\rangle + (-1)^{\mathrm{Ans}_2(y,Q_2(j,r))}|2,2,Q_2(j,r)\rangle\right) .$$

The significance of this quantum state is brought out in the following fact, implicit in the work of Kerenidis and de Wolf.

Fact 3.4.6 (Kerenidis and de Wolf [KdW03]). *By measuring* $|\phi_{jy}\rangle$ *in an appropriate basis, independent of* j *and* y*, followed by classical postprocessing, one can obtain a bit that equals* y_j *with probability at least* $11/14 - 4\varepsilon/7$.

To see how this fact can be used to obtain the desired communication protocol, note that

$$\begin{aligned}|\phi_{jy}\rangle &= \sum_{r\in\{0,1\}^\rho}\sum_{i=0}^{2}(-1)^{\mathrm{Ans}_i(y,Q_i(j,r))}c\,|r,i,i,Q_i(j,r)\rangle \\ &= \sum_{i=0}^{2}\sum_{z\in\{0,1\}^{\ell_q}}\sum_{r\in S_{ijz}}(-1)^{\mathrm{Ans}_i(y,z)}c\,|r,i,i,z\rangle \\ &= \sum_{i=0}^{2}\sum_{z\in\{0,1\}^{\ell_q}}|\chi_{ijz}\rangle\cdot(-1)^{\mathrm{Ans}_i(y,z)}c\sqrt{|S_{ijz}|}\,|i,z\rangle\,,\end{aligned}$$

where $|\chi_{ijz}\rangle := |S_{ijz}|^{-1/2}\sum_{r\in S_{ijz}}|r,i\rangle$. Let U_j be a unitary transformation that maps $|0^\rho,0,i,z\rangle$ to $|\chi_{ijz}\rangle|i,z\rangle$. The protocol for INDEX$_n$ works as follows. Alice, on input y, prepares the quantum state

$$|\psi_{jy}\rangle := \sum_{i=0}^{2}\sum_{z\in\{0,1\}^{\ell_q}}(-1)^{\mathrm{Ans}_i(y,z)}c\sqrt{|S_{ijz}|}\,|i,z\rangle \qquad (3.1)$$

and sends it to Bob. Although it seems at first glance that $|\psi_{jy}\rangle$ depends on j, it in fact does not, because the perfect privacy guarantee of P implies that for any $j,k\in[n]$,

$$\frac{|S_{ijz}|}{2^\rho} = \Pr_R[Q_i(j,R)=z] = \Pr_R[Q_i(k,R)=z] = \frac{|S_{ikz}|}{2^\rho}\,. \qquad (3.2)$$

Bob, upon receiving $|\psi_{jy}\rangle$, constructs the state $|0^\rho,0\rangle|\psi_{jy}\rangle$ using ρ qubits of his own and applies U_j to it, obtaining the state $|\phi_{jy}\rangle$. He then uses the procedure implied by Fact 3.4.6 to compute his output bit, which is correct with probability at least $11/14-4\varepsilon/7$. Since $|\psi_{jy}\rangle$ is a $(2+\ell_q)$-qubit state, the communication cost of this protocol is $2+\ell_q$. Fact 3.4.4 now implies that $\ell_q \geq (1-H(11/14-4\varepsilon/7))\,n-2$, giving us a lower bound on $\mathrm{cost}(P)$.

□

3.4.3 The nearly private 1-bit case

Without perfect privacy, the argument of the previous subsection does not work. This is because Equation (3.2) no longer holds, which makes the above quantum communication protocol ill-defined: Alice can no longer prepare the state $|\psi_{jy}\rangle$ because it *might* depend on j, which Alice does not know. However, we shall show that Alice can get away with sending Bob the state $|\psi_{1y}\rangle$, provided a sufficiently strong privacy guarantee holds.

Theorem 3.4.7. *Let ε and δ be sufficiently small positive reals. Then any 2-server $(\ell_q, 1; \varepsilon, \delta)$-PIR protocol has $\ell_q \geq (1 - H(3/4 + 2\delta/3 - \sqrt{2\delta} - \varepsilon))\, n$. In particular, the communication cost of such a protocol is at least $\Omega_{\varepsilon,\delta}(n)$.*

Proof. We use the framework and notation of Section 3.4.1. Suppose P is a 2-server $(\ell_q, 1; \varepsilon, \delta)$-PIR protocol. Consider the following one-way communication protocol for INDEX$_n$: Alice, on input y, sends Bob the $(2 + \ell_q)$-qubit quantum state $|\psi_{1y}\rangle$. Bob, upon receiving it, constructs the state $|0^\rho\rangle|\psi_{1y}\rangle$ defined by Eq. (3.1) and applies the unitary transformation U_j to it. He then measures the resulting state $|\phi'_{jy}\rangle$ as mentioned in Fact 3.4.6.

Examining the details of Kerenidis and de Wolf's argument, we see that, for each $r \in \{0,1\}^\rho$, measuring $|\phi_{jy}\rangle$ would have correctly obtained the pair of bits $(\mathrm{Ans}_1(y, Q_1(j,r)), \mathrm{Ans}_2(y, Q_2(j,r)))$ with probability at least $3/4$. Let us eschew the additional "11/14 trick" used in that argument and instead consider the probability p of obtaining this same outcome by measuring Bob's state $|\phi'_{jy}\rangle$. Let $|\mu\rangle$ be the vector in the measurement basis corresponding to the desired outcome, so that $|\langle\mu|\phi_{jy}\rangle|^2 \geq 3/4$. Then

$$|\langle\mu|\phi'_{jy}\rangle| \;\geq\; |\langle\mu|\phi_{jy}\rangle| - |\langle\mu|\phi_{jy} - \phi'_{jy}\rangle| \;\geq\; \frac{\sqrt{3}}{2} - \||\phi_{jy}\rangle - |\phi'_{jy}\rangle\|_2 \,. \tag{3.3}$$

Now,

$$\begin{aligned}
\||\phi_{jy}\rangle - |\phi'_{jy}\rangle\|_2^2 &= \||0^\rho, 0\rangle|\phi_{jy}\rangle - |0^\rho, 0\rangle|\phi'_{jy}\rangle\|_2^2 \\
&= \||\psi_{jy}\rangle - |\psi_{1y}\rangle\|_2^2 \qquad (3.4) \\
&= \sum_{i=0}^{2} \sum_{z \in \{0,1\}^{\ell_q}} c^2 \left(\sqrt{|S_{ijz}|} - \sqrt{|S_{i1z}|}\right)^2 \qquad (3.5) \\
&\leq \sum_{i=0}^{2} \sum_{z \in \{0,1\}^{\ell_q}} c^2 \big||S_{ijz}| - |S_{i1z}|\big|
\end{aligned}$$

where Eq. (3.4) holds because U_j is unitary, and Eq. (3.5) is obtained by invoking Eq. (3.1). Since P has privacy parameter δ, for $i \in \{1,2\}$ we have $\sum_{z\in\{0,1\}^\rho} ||S_{ijz}| - |S_{i1z}|| \leq 2^\rho \delta$. Also, by design, $S_{0jz} = S_{01z}$ for all z. Putting these facts together and using Eq. (3.3) gives

$$|\langle \mu | \phi'_{jy} \rangle|^2 \ \geq \ \left(\frac{\sqrt{3}}{2} - \sqrt{2c^2 2^\rho \delta} \right)^2 \ = \ \frac{3}{4} + \frac{2\delta}{3} - \sqrt{2\delta}\,.$$

Since Bob eschews the classical postprocessing (the "11/14 trick"), the probability that he correctly outputs y_j is at least the above quantity minus the probability that the PIR scheme errs, i.e., at least $3/4 + 2\delta/3 - \sqrt{2\delta} - \varepsilon$. The theorem follows. □

3.5 Nearly private bounds using other measures

3.5.1 Hellinger distance

Hellinger distance between two distributions with densities p and q over the space Ω with measure λ is defined as

$$D_H(p,q) = \sqrt{\int_\Omega |\sqrt{p(x)} - \sqrt{q(x)}| d\lambda(x)}.$$

The following bounds on it are known.

$$\frac{1}{2} D_H^2(p,q) \leq \|p-q\|_1 \leq D_H(p,q).$$

If we use Hellinger distance as a measure of similarity between distributions and declare that a private information scheme is nearly private with privacy parameter δ in Hellinger sense, this scheme will also be nearly private with privacy parameter δ in L_1 sense.

Therefore, if we use Hellinger distance instead of L_1 distance, all the negative results of this thesis will still hold.

3.5.2 Relative entropy

Relative entropy or Kullback-Leibler distance between two distributions with densities p and q over the space Ω with measure λ is defined as

$$D_{KL}(p\|q) = \int_\Omega p(x) \log \frac{p(x)}{q(x)} dx.$$

Pinsker's inequality states that

$$D_{KL}(p\|q) \geq \frac{1}{2 \ln 2} \|p - q\|_1^2.$$

Therefore, if we were to use Kullback-Leibler distance instead of L_1 distance, a scheme that is nearly private with privacy parameter δ in the Kullback-Leibler sense would also be private with privacy parameter $\sqrt{(2 \ln 2)\delta}$ in L_1 sense. The negative results of this chapter would hold if we substitute $\sqrt{(2 \ln 2)\delta}$ for δ.

3.5.3 Mutual information

Consider the mutual information I between the index j and the query $Q_i(j, r)$. If the information is 0, the server i knows nothing about the index j after getting the query, and therefore the scheme is perfectly private.

Consider the case of one server and perfect recovery, and let $I = \delta$. An easy upper bound can be achieved by writing down the index j in binary and transmitting δ first bits as a query. In order to guarantee perfect recovery, the server needs only to transmit bits at the indices starting with the query, therefore the total communication will be $\delta + n/2^\delta$ (as opposed to n in the perfectly private case). We conjecture that the lower bound is $n/2^\delta$.

Chapter 4

Conclusions and future work

In my graduate work I have shown a number of privacy-related results united by the common theme of allowing less than perfect privacy in schemes where formerly only or mostly perfect privacy was considered. The field of privacy schemes is full of possibilities for trade-offs that frequently allow getting useful properties.

We introduced web caches as a useful tool for anonymous web-browsing. Future work could include web caches that are specifically targetted towards this goal. Note that our present analysis concentrated on using existing caching systems. However, it is natural to pose a different question, namely: how should an anonymity-friendly caching system be designed from the ground up? In particular, an interesting direction of research would be methods for handling dynamic content without sacrificing too much user privacy. It would be interesting to do a user study on web-browsing patterns. Such a study may show that less protection is required for web browsing that involves dynamic content due to the high variability of requests for such content.

For example, it could be interesting to look at the published dataset of AOL search queries to see what amount of queries made by the same users could still be associated if the users' IPs were hidden, what amount of queries would be *personwhacks* - i.e., similarly to googlewhacks, allowing to uniquely identify the person they refer to. It is possible that merely hiding users' IPs from the search engine would significantly impair the ability of the search engine operator to correlate queries as belonging to the same user. In particular, hiding users' IPs may dissociate queries containing personally identifiable information from queries containing information that the users might want to conceal. Perhaps a caching proxy could be combined with another anonymizing system, for example Tor, to allow access to certain types of dynamic content without incurring a great loss of privacy. Here, as in numerous other cases, the composition of several security systems may exhibit unexpected properties not present in the components.

We produced a coercion-resistant voter verifiable electronic voting scheme. In the future this scheme could be made more practical and voter-friendly. It may be worth the effort (and certainly necessary if there is a possibility for ever getting the system used in the field) to attempt to give each component a crisp, intuitive, and user-friendly look. Currently, digests of cryptographic keys, such as ssh keys and PGP keys are presented to the user as long strings of characters; however, humans are not really comfortable with comparing (as well as processing or remembering) such strings. Various id numbers and signatures used in our scheme could benefit from being presented in a more user-friendly form. Trade-offs between usability and protection from possible attacks may also be worth exploring. It would also be interesting to look at extending the system to support different types of elections (whereas currently our scheme only allows the voter to choose one candidate). It would be interesting to conduct a user study to see if our scheme actually achieves voter trust - namely, if the non-technical users trying it out would have the impression that they can indeed verify that their vote made it to the tally.

We studied the generalization of private information retrieval to nearly private information retrieval and proved a number of new bounds on the amount of communication required in nearly private information retrieval schemes. We have found that in the 1-server case and in the binary 2-server case relaxing the privacy requirements on a private information retrieval (PIR) scheme by allowing it to leak a small amount of information about the client's index does not allow more than a constant factor improvement in the communication cost. The question of whether improvements can be obtained for the general 2-server case remains open. The generalization to nearly private information retrieval for the case of 3 or more servers so far also remains an entirely unexplored topic that may be worth addressing in the future. It would also be interesting to see whether schemes with small mutual information between the index and the corresponding query could be more efficient or whether the conjectured lower bound holds.

Bibliography

[Abl96] F. Ablayev. Lower bounds for one-way probabilistic communication complexity and their application to space complexity. *Theoretical Computer Science*, 175(2):139–159, 1996.

[BFG06] Richard Beigel, Lance Fortnow, and William Gasarch. A tight lower bound for restricted PIR protocols. *Computational Complexity*, 15(1):82–91, May 2006.

[BFK01] Oliver Berthold, Hannes Federrath, and Stefan Köpsell. Web MIXes: A system for anonymous and unobservable Internet access. *Lecture Notes in Computer Science*, 2009:115–??, 2001.

[BIM00] A. Beimel, Y. Ishai, and T. Malkin. Reducing the servers computation in private information retrieval: PIR with preprocessing. In *CRYPTO 2000, LNCS 1880*, pages 56–74. 2000.

[BL02] Oliver Berthold and Heinrich Langos. Dummy traffic against long term intersection attacks. In *Privacy Enhancing Technologies 2002*. Springer-Verlag, 2002.

[BP98] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual search engine. In *Seventh International World Wide Web Conference*, 1998.

[BR94] M. Bellare and P. Rogaway. Optimal asymmetric encryption/how to encrypt with RSA. In *Advances in Cryptology—Eurocrypt '94*, volume 950, pages 92–111, 1994.

[BT94] Josh Benaloh and Dwight Tuinstra. Receipt-free secret-ballot elections. In *Proc. of 26th Symp. on Theory of Computing (STOC'94)*, pages 544–553, New York, 1994.

[CD03] Tara Calishain and Rael Dornfest. *Google hacks*. O'Reilly, 2003.

[CG97] Benny Chor and Niv Gilboa. Computationally private information retrieval (extended abstract). pages 304–313, 1997.

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. In *IEEE Symposium on Foundations of Computer Science*, pages 41–50, 1995.

[CGKS98] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private information retrieval. *J. ACM*, 45(6):965–982, 1998.

[Cha81] David Chaum. Untraceable electronic mail, return addresses and digital pseudonyms. *Communications of the ACM*, 24(2):84–88, February 1981.

[Cha04] David Chaum. Secret-ballot receipts: true voter-verifiable elections. *IEEE Security & Privacy*, 2(1):38–47, January/February 2004.

[Com] Comet. Comet web search. `http://search.cometsystems.com`.

[CS07] Amit Chakrabarti and Anna Shubina. Nearly private information retrieval. In *MFCS*, pages 383–393, 2007.

[Dai98] Wei Dai. PipeNet 1.1. `http://www.eskimo.com/~weidai/pipenet.txt`, 1998.

[DAR03] DARPA. Report to Congress regarding the Terrorism Information Awareness program. `http://www.darpa.mil/body/tia/tia_report_page.htm`, May 2003.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: the second-generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[Ele97] Electronic Privacy Information Center. Surfer beware: personal privacy and the Internet. `http://www.epic.org/reports/surfer-beware.html`, June 1997.

[Ele03] Electronic Privacy Information Center. Privacy and consumer profiling. `http://www.epic.org/privacy/profiling`, April 2003.

[FM02] Michael J. Freedman and Robert Morris. Tarzan: a peer-to-peer anonymizing network layer. In *Proceedings of the 9th ACM Conference on Computer and Communications Security (CCS 2002)*, Washington, D.C., November 2002.

[Gas04] W. Gasarch. A survey on private information retrieval, 2004.

[Gig] Gigablast. Gigablast search. `http://gigablast.com`.

[GKST02] O. Goldreich, H. Karloff, L. Schulman, and L. Trevisan. Lower bounds for linear local decodable codes and private information retrieval systems. In *Proceedings of the 17th IEEE Conference on Computational Complexity. IEEE Computer Society Press*, pages 175–183, 2002.

[Goo] Google. Google. `http://www.google.com`.

[GS01] Ian Goldberg and Adam Shostack. Freedom network 1.0 architecture and protocols, October 2001.

[HS00] Martin Hirt and Kazue Sako. Efficient receipt-free voting based on homomorphic encryption. *Lecture Notes in Computer Science*, 1807:539–??, 2000.

[JJ] Ari Juels and Markus Jakobsson. Coercion-resistant electronic elections.

[JVZ01] Shu Jiang, Nitin H. Vaidya, and Wei Zhao. Power-aware traffic cover mode to prevent traffic analysis in wireless ad hoc networks. In *IEEE Infocom*, 2001.

[KdW03] I. Kerenidis and R. de Wolf. Exponential lower bound for 2-query locally decodable codes. In *35th Annual ACM Symposium on Theory of Computing*, pages 106–115, 2003.

[KO97] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *IEEE Symposium on Foundations of Computer Science*, pages 364–373, 1997.

[KO00] Eyal Kushilevitz and Rafail Ostrovsky. One-way trapdoor permutations are sufficient for non-trivial single-server private information retrieval. *Lecture Notes in Computer Science*, 1807:104–121, 2000.

[LW05] Joseph K. Liu and Duncan S. Wong. A restricted multi-show credential system and its application on e-voting. In *ISPEC*, pages 268–279, 2005.

[LWW04] J. Liu, V. Wei, and D. Wong. Linkable spontaneous anonymous group signature for ad hoc groups, 2004.

[Man98] E. Mann. Private access to distributed information, 1998.

[MD05] Steven J. Murdoch and George Danezis. Low-cost traffic analysis of Tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*. IEEE CS, May 2005.

[Mer01] Rebecca Mercuri. Rebecca Mercuri's statement on electronic voting. `http://www.notablesoftware.com/RMstatement.html`, 2001.

[Mer02] Rebecca Mercuri. A better ballot box?, October 2002.

[MS02] David Martin and Andrew Schulman. Deanonymizing users of the SafeWeb anonymizing service. Technical Report 2002-003, 11 2002.

[NA03] Andrew Neff and Jim Adler. Verifiable e-Voting. `www.votehere.net/vhti/documentation/verifiable_e-voting.pdf`, 2003.

[Nay99] Ashwin Nayak. Optimal lower bounds for quantum automata and random access codes. In *IEEE Symposium on Foundations of Computer Science*, pages 369–377, 1999.

[NC00] Michael A. Nielsen and Isaac L. Chuang. *Quantum computation and quantum information*. Cambridge University Press, 2000.

[Oka96] Tatsuaki Okamoto. An electronic voting scheme. In *IFIP'96, Advanced IT Tools*, pages 21–30. Chapman & Hall, 1996.

[Oka97] Tatsuaki Okamoto. Receipt-free electronic voting schemes for large scale elections. *Security Protocols Workshop*, pages 25–35, 1997.

[Ols03a] Stefanie Olsen. Court backs thumbnail image linking. `http://news.com.com/2100-1025_3-1023629.html?tag=bplst`, 2003.

[Ols03b] Stefanie Olsen. Google cache raises copyright concerns. `http://news.com.com/2100-1032_3-1024234.html?tag=fd_lede2_hed`, 2003.

[PK00] Andreas Pfitzmann and Marit Köhntopp. Anonymity, unobservability, and pseudonymity — a proposal for terminology. In *Designing Privacy Enhancing Technologies: Proceedings of the International Workshop on the Design Issues in Anonymity and Observability*, volume 2009, pages 1–9. Springer-Verlag, July 2000.

[Pou03] Kevin Poulsen. DirecTV dragnet snares innocent techies. `http://www.securityfocus.net/news/6402`, July 2003.

[Ray01] Jean-François Raymond. Traffic analysis: Protocols, attacks, design issues and open problems. In H. Federrath, editor, *Designing Privacy Enhancing Technologies: Proceedings of International Workshop on Design Issues in Anonymity and Unobservability*, volume 2009 of *LNCS*, pages 10–29. Springer-Verlag, 2001.

[Riv06] Ron Rivest. The ThreeBallot voting system. Technical report, 2006.

[RR98] Michael K. Reiter and Aviel D. Rubin. Crowds: anonymity for Web transactions. *ACM Transactions on Information and System Security*, 1(1):66–92, 1998.

[RS07] Ronald L. Rivest and Warren D. Smith. ThreeVotingProtocols: ThreeBallot, VAV, and Twin. In *Proceedings of EVT '07*, Boston, MA, August 2007.

[RST01] Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. *Lecture Notes in Computer Science*, 2248:552–565, 2001.

[RY06] A. Razborov and S. Yekhanin. An $\Omega(n^{1/3})$ lower bound for bilinear group based private information retrieval. In *47th Annual IEEE Symposium on Foundations of Computer Science*, pages 739–748, 2006.

[SC07] Radu Sion and Bogdan Carbunar. On the computational practicality of private information retrieval. *NDSS*, 2007.

[SG00] Adam Shostack and Ian Goldberg. How not to design a privacy system: reflections on the process behind the Freedom product. In *Proceedings of the tenth conference on computers, freedom and privacy: challenging the assumptions*, pages 85–87, Toronto, Ontario, Canada, 2000. ACM Press.

[SGR97] P.F. Syverson, D.M. Goldschlag, and M.G. Reed. Anonymous connections and Onion Routing. In *IEEE Symposium on Security and Privacy*, pages 44–54, Oakland, California, 4–7 1997.

[SK95] Kazue Sako and Joe Kilian. Receipt-free mix-type voting scheme—a practical solution to the implementation of a voting booth. In *Advances in Cryptology—Eurocrypt '95*, volume 921, pages 393–403, Berlin, 1995. Springer-Verlag.

[SK02] Ronggong Song and Larry Korba. Review of network-based approaches for privacy. In *Proceedings of the 14th Annual Canadian Information Technology Security Symposium*, Ottawa, Ontario, May 2002.

[SS03] Anna Shubina and Sean Smith. Using caching for browsing anonymity. In *ACM SIGEcom Exchanges*, volume 4. September 2003.

[SS04] Anna Shubina and Sean Smith. Design and prototype of a coercion-resistant, voter verifiable electronic voting system. In *Second Annual Conference on Privacy, Security and Trust*, pages 29–39, 2004.

[Ste98] Julien P. Stern. A new and efficient all-or-nothing disclosure of secrets protocol. In *ASIACRYPT: Advances in Cryptology – ASIACRYPT: International Conference on the Theory and Application of Cryptology*. LNCS, Springer-Verlag, 1998.

[The03] The Virtual Chase. Don't use the Google cache as anonymous surf. `http://www.virtualchase.com/ask_answer/google_cache.html`, May 2003.

[TW05] Patrick P. Tsang and Victor K. Wei. Short linkable ring signatures for e-voting, e-cash and attestation. In *ISPEC*, pages 48–60, 2005.

[Vot] VoteHere. VoteHere. `www.votehere.net`.

[Way] WayBack. WayBack Machine. `http://www.archive.org`.

[WY05] D. Woodruff and S. Yekhanin. A geometric approach to information theoretic private information retrieval. In *20th Annual IEEE Conference on Computational Complexity*, pages 275–284, 2005.

[Yek07] S. Yekhanin. Towards 3-query locally decodable codes of subexponential length. In *39th Annual ACM Symposium on the Theory of Computing*, 2007.