Metadata-Hiding Anonymous Communication with Authenticated Reading and Writing

Simon Beyzerov[†]

Mentor: Sacha Servan-Schreiber[‡]

Massachusetts Academy of Math & Science

Advanced STEM with Scientific and Technical Writing

Instructor: Kevin Crowthers, Ph.D. [†]

Worcester, MA. 01605

[†] Massachusetts Academy of Math & Science

[‡] MIT

Abstract

In this work, we present a set of lightweight constructions for anonymous communication with cryptographic privacy guarantees for users. Existing metadata-hiding messaging systems such as Express and Riposte are built in the sender-anonymous model, providing anonymity protections to clients attempting to write to a database of mailboxes or bulletins. These systems provide ad-hoc protections against malicious clients but fail to secure stored messages under a compromised database of secrets. This work improves and extends the familiar model of distributed trust leveraged in both Express and Riposte in several ways: First, a compact method of anonymous authentication is applied that hardens these systems to compromised user secrets. Finally, a new set of protocols is introduced that allows for anonymously authenticated client writing *and* reading simultaneously. Operating under a similar threat model, these improved systems are shown to scale to millions of users while shifting the privacy guarantees seen in the state of the art. Compared to Express, these techniques reduce latency, increase message throughput, and allow for over 100× faster client-side processing, all while improving security under passive adversaries. With increased efficiency and decreased communication costs for clients, these schemes are shown to be more practical in real-world deployment scenarios where clients may have limited computational resources. It has become abundantly clear that service providers and state-funded intelligence organizations cannot be trusted when it comes to protecting the personal data of users. Providing cryptographic guarantees at the system level, the protocols presented in this work introduce privacy by masking the powerful metadata underlying every user interaction.

Keywords: privacy, communication, anonymous, authentication, online messaging, cryptography, distributed point functions

Acknowledgements

I would like to thank Sacha Servan-Schreiber for his continued support throughout this project. His mentorship, encouragement, and review of the techniques presented in this work have been invaluable. I additionally would like to thank Dr. Crowthers for his extended guidance and support throughout the research process.

Metadata-Hiding Anonymous Communication with Authenticated Reading and Writing

Paired with the rise of consumer data analytics, the continued transition of interpersonal communication onto the Internet has made privacy an increasingly important requirement in many network systems. Yet the inherent structure of the Internet has (and continues to) place user privacy as a secondary consideration. Protecting users must start at the core of such systems and place little trust in service providers, whose actions have been shown to be easily swayed. Although attempts by services such as WhatsApp, iMessage, and Facebook Messenger to employ methods of end-to-end encryption have succeeded in securing message content, leaked *metadata* has exposed the power of underlying communication data. In general, metadata continues to be monitored by governments and independent parties as both a means of surveillance and monetization, typically under little legal restriction (Mayer et al., 2016; Schneier, 2014). Recently, cryptographic techniques to secure such metadata have been proposed, but questions remain about whether such systems can be practically deployed under realistic computational and security assumptions.

Background

Privacy

The internet's growing role in people's lives has made monumental impacts on how privacy and security are perceived. While formulating privacy's significance is challenging, it is important to recognize that privacy's fundamental ties to personal liberties stretch beyond the bounds of "normal" human interactions (Rachels, 1975). This sweeping role of privacy is made apparent when considering the internet's role in people's lives; digitized entertainment, communication, and education demand considerations of privacy, much like their physical equivalents.

The potential for exceptional protections within online mediums is undeniable, though the technical challenges with maintaining privacy through these paradigms must be recognized. Further, recognizing the unprecedented threats to individual autonomy is of key importance, and it starts by considering what aspects of privacy internet users are willing to surrender. Today, while most Americans are willing to sacrifice privacy for perceived benefits, there is a distinct line between what seems appropriate or inappropriate for outside parties to obtain (Rainie & Duggan, 2016).

Metadata

Online, one of the most important aspects of a user's privacy lies in their *metadata*. Broadly, metadata is any underlying information describing some object or data content (Dempsey & Heery, 1998). Its power lies in its ability to deduce some details without having direct access or knowledge of the object itself.

Metadata may be structured to some degree for actionable use (Riley, 2017). One example can be seen in the sorting system of modern file databases: files hold labeled metadata surrounding the date of a file's creation, modification, size, and its owner. This allows for quicker access by allowing users to sift through a large dataset with only contextual information.

Within the distributed model of the internet, metadata is even more pervasive. Online, every interaction, independent of its nature, holds some form of metadata– take for example the email address, IP address, and time of day revealed while accessing today's most popular online services. Yet, while metadata's role is crucial in ensuring information is transported efficiently, the recent rise in large data aggregation has given it new power; massive social media platforms such as Facebook have been found to harvest, sell, and misplace millions of users' metadata (Isaak & Hanna, 2018). These troves of data, paired with advances in both AI and data analysis, continue to be used for user manipulation, general surveillance, and identification (Perez et al., 2018).

Unfortunately, many users are unconscious of the invasive nature of such analyses. Metadata can reveal an immense amount about an individual without needing any direct disclosure of sensitive information. From Twitter metadata alone, it's possible to deduce gender, age, regional origin (Rao et al., 2010), emotion (Bollen et al., 2011), political alignment (Conover et al., 2011), and general identification (Perez et al., 2018). Within communication protocols, arguably the Internet's most fundamental primitive, metadata is closely tied to individual autonomy and demands further consideration. After all, the basis of the internet lay in improving connectivity among its users.

Communication

Increasingly, threats of surveillance, censorship, and the selling of user data by both service providers (Isaak & Hanna, 2018) and state-sponsored intelligence organizations (Greenwald, 2014) have underscored the need for privacy protections in online communication. Metadata within online communication systems can be observed in various contexts, ranging from the data associated with an individual message to the information about the users communicating. Accordingly, the specific form of the associated communication heavily influences how user privacy can be mishandled.

Whistleblowing is a relevant practical challenge that encapsulates the power of metadata. When considering whistleblowing tools such as SecureDrop (Berret, 2016), a natural setting may involve an individual attempting to communicate some sensitive message to another party through some central server. Implicitly, this central party gains valuable metadata surrounding the conversation between users: the time, date, sender, receiver, size, and other relevant information is revealed.

While the setting of whistleblowing demands stringent protections for both dissidents and

journalists, it's important to recognize that protecting metadata is increasingly relevant to average internet users, whether that be within social networks or dedicated communication platforms.

Protecting Metadata

With most traditional centralized communication systems, messages are stored on an intermediary server prior to being retrieved by users, where additional protection may be sustained. Transport-layer encryption (such as TLS) is instantiated in most modern infrastructure when writing or retrieving a message from the communication server (Dierks & Rescorla, 2008; Rescorla & Dierks, 2018). Although encrypting messages during transport secures content from *man-in-the-middle attacks* that may attempt to intercept data, it provides no guarantees against the intermediary server(s) that store and maintain user messages—these service providers may freely access any information and metadata associated with the interaction.

Traditional techniques for achieving some notion of privacy from intermediaries start with end-to-end encryption (E2EE), with many modern applications adopting it as a means of securing content (Endeley, 2018). E2EE considers a public-key model that ensures only the recipient of a message can safely decrypt its contents, thus rendering raw message contents useless to a malicious service provider. While popular services like WhatsApp, iMessage, and Facebook Messenger have adopted E2EE, questions remain surrounding the metadata that cannot be trivially encrypted (Bai et al., 2020; Lee et al., 2017).

Encryption, though non-conducive to any meaningful masking of metadata, works well as an initial step towards securing message content. With encryption alone, service providers still learn *who* a client is communicating with. Techniques involving *private information retrieval* (PIR) have been shown to deal with such an issue: any sender can privately "select" a user to write to, without the central messaging server learning which user's "mailbox" was modified (Chor et al., 1995).

Metadata-hiding communication

Dealing with metadata leakage in online communication has been examined by a host of systems, typically considering either *differential* or *cryptographic* privacy guarantees (Abraham et al., 2020; Corrigan-Gibbs et al., 2015; Eskandarian et al., 2021; Newman et al., 2021; Vadapalli et al., 2021). Differentially private systems are typically more performant, but their privacy degrades with every invocation. Cryptographic schemes, albeit at a cost, ensure no data leakage, even over a large number of interactions. While reliance on heavy cryptographic primitives had made these systems too difficult to deploy in practice, recent exposition in systems like Express (Eskandarian et al., 2021) has firmly established the potential practicality of such protocols.

Many of these works typically consider a sender-anonymous model of privacy, which provably

anonymizes client *writing* of messages and ensures little metadata about the underlying message content can be deduced. Recent cryptographic schemes, such as Express, achieve this through a Private Information Retrieval (PIR) approach (Eskandarian et al., 2021). The foregoing privacy model typically necessitates some form of robustness against malicious clients— a client masking both their identity and message content seems capable of corrupting system availability or individual communications and directly opening a gate for censorship. Service providers must somehow filter out malicious clients from a collection of anonymous requests.

Accordingly, ensuring proper access control to these systems is typically dealt with implicitly or in an ad-hoc manner. In the PIR model of Express, virtual addresses are used as a method of authentication: only a user with knowledge of a set random value– the virtual address– can successfully write to another user's mailbox (Eskandarian et al., 2021). Unfortunately, these systems require such addresses to be stored in the clear. This opens up Express to a potential security concern: any passive adversary who gains access to the database of virtual addresses and decryption keys can gain free rein to manipulate any mailbox.

These existing limitations prompt a set of lingering questions:

1. Can cryptographic guarantees be enforced while providing both anonymous writing and reading?
2. Can this be achieved with a stronger notion of robustness against malicious clients?
3. Most importantly, can this be achieved without imposing impractical client-side computation overheads?

Objectives

This work aims to answer these questions in the affirmative by introducing a set of cryptographic protocols for metadata-hiding anonymous communication. We first build upon the distributed model presented by Eskandarian et al. (2021) by adopting the PIR-based flow of message access. Using *Distributed Point Functions* along with secret-shared zero-knowledge proofs of knowledge, we are able to harden previous works such as Express while providing concrete efficiency improvements. We then aim to illustrate that our approach can be generalized to provide both anonymous client writing and reading under a modified threat model. Crucially, we seek to introduce the notion of *anonymous authentication* which ensures communications are cryptographically private and are secure against both malicious clients and service providers.

Preliminaries

Prior to presenting protocols for achieving metadata-hiding anonymous authentication, we will outline a set of existing cryptographic primitives. These will be crucial in providing intuition behind our

constructions and introducing a loose mathematical notion of privacy, security, and anonymity. Please consider the referenced appendices which provide additional details and formalized definitions, as black-box descriptions of these primitives are almost exclusively utilized.

Role of Student v. Mentor

The following work was completed over the course of approximately 6 months, under the mentorship of Sacha Servan-Schreiber. The student worked on the entirety of the project while receiving occasional advice from the mentor, who aided in the planning process and provided guidance at the project's conclusion. Additionally, some aspects of the code and evaluations build off a library that was created collaboratively between the student and the mentor during a previous project. This previous work is currently unpublished and is cited throughout this work as a utilized method— prior to publication, we cite it as a personal communication between the mentor and the student: (S. Servan-Schreiber, personal communication, February 1, 2022).

Distributed Trust

Distributed primitives are a natural (and widely explored) approach for achieving cryptographic privacy, particularly when dealing with private information. Under carefully formalized assumptions, allowing information to be shared amongst multiple parties can introduce immense privacy, security, and efficiency benefits unattainable with only one party (Beimel & Ishai, 2001; Goldwasser, 1997).

Secret Sharing

When attempting to keep a piece of information secret, it can be helpful to split it into many pieces. Secret sharing is a powerful primitive that allows a user to share a secret amongst l parties. Crucially, these l parties learn nothing about the user's data unless some fixed threshold $t \leq l$ comes together to reconstruct the secret.

Additive secret sharing is one common variant of Secret Sharing that uses the properties of addition in a finite field. This is an example of general linear secret sharing (LSS), where computing shares must be done with linear operations (Karnin et al., 1983). For a secret $s \in \mathbb{F}$, let $([s]_1, \ldots, [s]_l)$ denote the set of additive secret shares distributed amongst a set of l parties. If $t = l = 2$, it's intuitive to imagine the following additive LSS scheme: one party receives a random integer, which is their share $[s]_1$. The other receives $[s]_2 = s - [s]_1 \in \mathbb{F}$. On their own, these two shares appear completely random and reveal nothing about the value of s. When added together, they reveal the secret value.

Within this work, a mix of additive and XOR-based secret sharing is utilized (Wang, 2015; Wang &

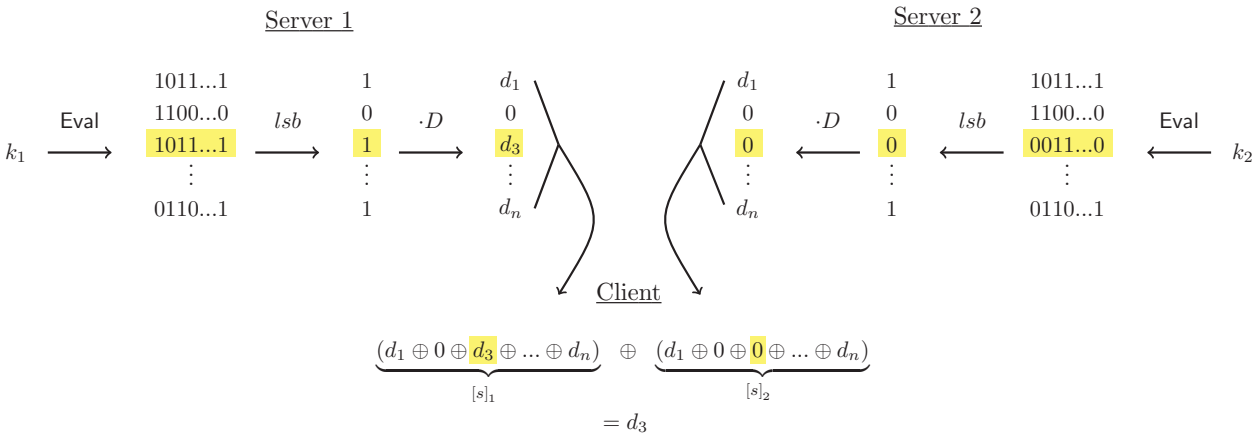


Figure 1
Overview of two-server XOR-based PIR-reading. A pair of servers start with DPF keys $k_{1,2}$, which they evaluate and calculate a cumulative XOR value based on the result of the DPF expansion. These cumulative values are sent to the client, who can combine them with XOR to retrieve the database row. If a user wishes to read row i, the two servers' expansions will be identical at all indices, except index i. To the servers, this distribution appears indistinguishable from random (Chor et al., 1995).

Desmedt, 2014). In this secret sharing scheme, shares are not added, but rather combined with bit-wise XOR. For our previous example with two parties, instead of having $s = [s]_1 + [s]_2$, we have $s = [s]_1 \oplus [s]_2$. Observe that no one bit from a known secret share reveals any information about a bit in the corresponding secret share held by another party— bit-wise XOR is equivalent to addition in $\mathbb{Z}_2$.

Private Information Retrieval (PIR)

Although secret sharing allows for the storage of private data amongst a set of servers, a meaningful question arises in how such data can be securely and privately retrieved. Beyond simply requesting some data from a server, a user may wish to do so privately, not revealing *which* piece of data was selected.

A wide array of schemes for Private Information Retrieval (PIR) have been presented, both under computational[1] and information-theoretic[2] privacy assumptions (Aguilar-Melchor et al., 2015; Beimel et al., 2005). Under information-theoretic privacy, it is useful to consider multi-server PIR schemes, which allow a user to retrieve some data d_i from a database $D = (d_1, \ldots, d_n)$ of size n that is distributed among l servers. Crucially, so long as the l servers do not maliciously cooperate (collude), they cannot learn anything about the data being retrieved, even under unlimited computational resources.

One intuitive method of achieving information-theoretic PIR has a database $D = (d_1, \ldots, d_n)$ distributed among a set of l servers, where each row value d resides in a finite field $\mathbb{F}_p$. A user attempting to

[1] Assumes adversaries have some bounded amount of computation (e.g., probabilistic polynomial-time)

[2] Assumes adversaries are computationally unbounded

retrieve row i can then provide each server with a secret-shared one-hot vector $S = ([0], \ldots, [1], \ldots, [0])$. This vector of secret-shares combines to a vector where each entry is equal to 0, except at position i, where it is equal to 1. By the properties of secret sharing, they do not learn of the binary value at any index, unless every server combines its share. With additive secret sharing, a useful property arises when these shares are multiplied against their corresponding database row. Note that $\sum d \cdot [0] = 0$ if the shares add to 0, and $\sum d \cdot [1] = d$ if the shares add to 1. By calculating an inner-product between the database values and the share of the one-hot vector, a server can compute a secret share of the desired database row: $d_1 \cdot [0] + d_2 \cdot [0] + \ldots + d_i \cdot [1] + \ldots = [d_i]$. Since the share of each row appears random, this final share also is indistinguishable from random. Servers can then send their respective shares to the user, who can combine them to retrieve d_i. We leverage an analogous approach with XOR shares within our constructions.

Distributed Point Functions (DPFs)

Although the previously outlined method for PIR is information-theoretically private, the communication costs incurred under this solution make it infeasible for any real-world system: the client must send each server a secret-shared vector that is *linear* to the size of the database (one additive secret share for each database row).

First presented by Boyle et al. (2016), Distributed point functions (DPFs) are a useful cryptographic technique that allows for the communication overhead of traditional multi-server PIR to be reduced to a logarithmic factor in the size of the database. Instead of sending $\mathcal{O}(n)$ bits of data, clients are able to compress this exchange to only $\mathcal{O}(\log n)$ bits, where n is the size of the database. Crucially, this compression ensures *sublinear* (and optimal) communication costs for clients without sacrificing any of the key properties of multi-server PIR.[3]

DPFs achieve such optimizations by encoding client queries as a *point function*. A point function $P_{\alpha,\beta}$ is a function that evaluates to α on input β and evaluates to 0 on all other inputs. This point function can be used to compress PIR, where a one-hot vector encodes a series of 0 values and a single non-zero index.

Clients encode their desired index i into two *keys* which can individually be *evaluated* by a server. When a server evaluates a DPF key k on some index, the value they output will be either 0 or α. Crucially, with DPFs this value is secret-shared, so neither server learns the exact output of the query.

DPFs consist of two algorithms, Gen and Verify, which define the structure of this interaction. First, clients run Gen, outputting two (or more) DPF keys, with each server receiving one such key. Upon receiving a key, servers can run Eval on any input to evaluate the point function at that value. This outputs an

[3] By nature of the construction of DPFs, this operation is *computationally* secure, rather than perfectly information-theoretic (Boyle et al., 2016)

additive secret share of the point function at that input. If the input happens to be β, the sum of all Eval operations will add to α. Otherwise, this sum will be 0.

Verifiable DPFs (VDPFs)

A key limitation of DPFs for PIR lies precisely in their strong privacy guarantees: for certain applications, it is not immediately apparent how servers can ensure a client strictly sets $\alpha = 1$. In fact, there is nothing stopping a client from not sharing a point function at all. For reasons that will become apparent in the upcoming section on Anonymous Authentication, this is an increasingly important issue that can open critical security flaws.

To ameliorate such a limitation, de Castro and Polychroniadou propose Verifiable Distributed Point Functions (VDPFs), a twist on DPFs that enforces *at most* one non-zero element within the encoded function (de Castro & Polychroniadou, 2021). Although this does not quite imply a properly formed point function, a simple modification can be performed to achieve this fact; it is possible to enforce exactly one entry of 1 encoded by the VDPF (S. Servan-Schreiber, personal communication, February 1, 2022).

VDPFs consist of three algorithms, Gen, Eval, and Verify. All but the latter hold the same fundamental functionality as those of DPFs. To ensure a client's VDPF encodes a one-hot vector, servers run Verify amongst themselves following the evaluation of their respective VDPF keys. This verification protocol checks to make sure the client query is well-formed. Unlike general DPFs, VDPFs are built exclusively for two server PIR and do not generalize to more parties.

Anonymous Authentication

While the ability to compress a PIR query to a size logarithmic in the number of mailbox rows solves the issue of high client costs, a challenge remains in addressing potentially malicious requests. Suppose a client wishes to write to another mailbox. While the aforementioned primitives ensure their request selects at most one mailbox, there is nothing preventing them from editing an arbitrary mailbox. This is crucial when considering scenarios where mailboxes may hold highly personal data, such as the communication scenario considered in this work.

Discrete Logarithm Problem

The computational intractability of the Discrete Logarithm Problem (DLP) is a fundamental assumption in complexity theory and is the basis of modern cryptography (Diffie & Hellman, 1976; ElGamal, 1985; Shor, 1999). For given values a, b, DLP considers the difficulty of computing the logarithm $x = \log_b a$. DLP is specifically concerned with this operation as defined in a group $\mathbb{G}$, rather than $\mathbb{R}$.

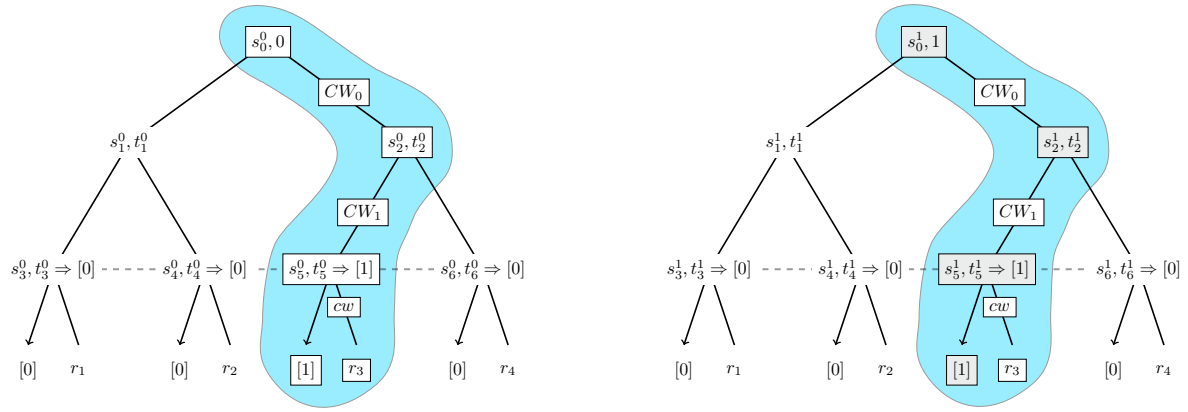


Figure 2

Binary tree representations of both parties' views of evaluating a VDPF, as presented by de Castro and Polychroniadou (2021). Following a Goldreich-Goldwasser-Micali construction, a VDPF considers a secret-shared binary tree with a punctured path (the special path). A left branch corresponds with a binary 1, whereas a right branch corresponds with 0. Each party's tree has identical nodes, except along the special path that describes α (recall that a VDPF will evaluate to 0 at all inputs, except on input α, where the output will be β). A (V)DPF maintains the invariant that $s_i^0 = s_i^1$ and $t_i^0 = t_i^1$ while a computed node is not on the special path, and otherwise $s_i^0 \neq s_i^1$ and $t_i^0 \neq t_i^1$. Servers use correction words to maintain this invariant. A VDPF checks an that an encoded point function is well-formed by adding an additional layer, encoding a value r that must match on both servers. Since correction words can only change one move down the tree, a well formed DPF must be correct up to the penultimate tree layer for verification to pass.

As utilized in this work, we consider the following formulation of DLP: Under the multiplicative cyclic group $\mathbb{Z}_p^*$ with generator g and a value h, it is conjectured that there does not exist a polynomial-time algorithm that can efficiently extract x, where $h = g^x \bmod p$. Presently, p must be a large Sophie Germain prime, typically 2048 bits (Pohlig & Hellman, 1978).

Cloak & SPoSS

Presented by Servan-Schreiber et al. (2022), a distributed discrete logarithm zero-knowledge proof of knowledge has been presented. Within the presented SPoSS proof, a client is able to prove in *zero-knowledge* that they possess the discrete logarithm of a secret-shared field element. With zero-knowledge, this scheme guarantees that the client need not reveal this value of x, nor the value g^x to the verifiers. By pairing such a proof with VDPFs, the Cloak system achieves a formalized authentication procedure with public-key storage— storing the values of g^x by the verifier does not reveal anything about each secret value x. Much like a password-hashing scheme, this structure ensures that knowledge of such public key g^x is insufficient to proper proof of knowledge of x.

Overview

The following section outlines the structure and properties of multiple constructions for metadata-hiding anonymous communication. These protocols build on the celebrated schemes of Express and its predecessors while utilizing a different approach to deal with some of their limitations.

Sender-anonymous Communication

In many settings, achieving anonymity for client writes is sufficient for user privacy. In the sender-anonymous model, a set of $l \geq 2$ servers host a set of "mailboxes." Clients holding proper access privileges are able to contribute a mailbox with an arbitrary "message," typically represented by some bounded integer value. Crucially, clients disclose neither the mailbox they chose to write to, nor the content of their message.

Threat Model

In this work, we consider a multi-server communication setting with an arbitrary amount of interacting clients. Our underlying construction considers the two-server sender-anonymous model, with all clients being fully untrusted and servers assumed to deviate from protocol in order to identify clients, decrypt messages, and link write requests. So long as these servers do not actively collude, they are free to act maliciously. We note that, much like Express, protecting against impersonation attacks— where one malicious server impersonates a client— is not offered by default. Clients can provide unique decryption keys in order to achieve message security in such a setting.

Anonymity is achieved through the notions of unlinkability and indistinguishability: every write request cannot be linked to any one user and nothing can be gleaned surrounding the recipient of any message. Concretely, this ensures the servers do not learn *who* performed a write request and cannot identify to *whom* it was addressed. Within the sender-anonymous model, no guarantees are made regarding read requests, which require a user to identify themselves in order to read a mailbox— this is the setting considered within Construction 1 and Express. As we build upon a private read and write scheme, anonymity is expanded analogously to read requests: the servers do not gain any information regarding *which* user read *which* mailbox. We make no anonymity guarantees for any external tracking (e.g., based on a user's IP address or other network data). This can be addressed with dedicated anonymous browsing systems (Dingledine et al., 2004).

For both the sender-anonymous and private read & write schemes, we guarantee that client anonymity is preserved, so long as the servers do not collude. Even over many iterations, servers cannot gain any information about the nature of user interactions and communications. Malicious servers additionally

cannot glean any data surrounding a client's message: the message encryption is indistinguishable from random. Finally, by the properties of the anonymous authentication we utilize, no malicious client can disrupt availability for other users (S. Servan-Schreiber, personal communication, February 1, 2022). This is in line with the guarantee that only users with proper access rights to a specific mailbox can interact with that mailbox.

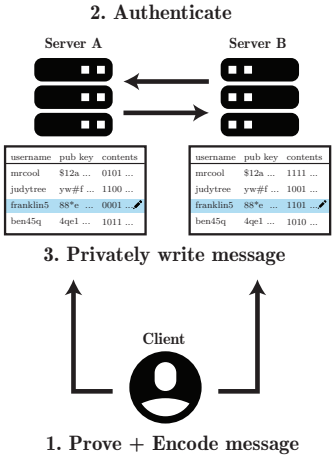
Express

Express, the celebrated work of Eskandarian et al. (2021), represents the current state-of-the-art in sender-anonymous systems and operates in a two-server setting. At registration, mailbox owners receive a randomized λ-bit *virtual address* that uniquely "unlocks" their mailbox, where λ is a statistical security parameter (Express considers $\lambda = 128$). When attempting to read messages, clients send this address, along with the index of the mailbox they hope to access. Servers check that the pair matches their records and send the associated contents, prior to emptying the mailbox. It is crucial to understand that the servers still learn which mailbox a user read from. Additionally, note that each server does not hold the mailbox contents in the clear. Instead, they each hold *additive* secret shares that, when combined, reveal the message. This means systems must operate under a non-collusion assumption to maintain privacy, as the servers can otherwise reveal messages following each client write.

To anonymously write a message, a client must hold the associated virtual address to the mailbox—distributing this address is assumed to be performed by some out-of-band means. Clients encode the message m into a pair of DPF keys that are each provided to the servers. These DPF keys evaluate to m when the specific virtual address is the input. Express servers then effectuate the write request by adding the result of every DPF evaluation over the set of virtual addresses. This operation can be seen as a PIR-write request and adds 0 to every database row other than the one with the matching address.

While virtual addresses ensure only certain mailboxes can be selected, by themselves they do not protect Express from disruptions by malicious clients. If a pair of received DPF keys do not encode a point function, clients are free to write to as many mailboxes as they wish, regardless of their access rights. Eskandarian et al. (2021) deal with this by having clients send a *secret-shared non-interactive proof* (SNIP), proving that the DPF keys encode *at most* one positive value. This *audit* protocol is then verified by the servers prior to writing. Crucially, this solution only protects against malformed DPFs, and does not consider whether the unique address accessed by the user is actually owned by that user: virtual addressing is the implicit solution offered by Express (using the properties of DPFs).

Express, although successful in attaining the security and privacy properties required for a sender-anonymous system, suffers from a set of fundamental limitations that inhibit its viability within



Client.Write(m, u_i, x_i)

1 : $(k_A, k_B) \leftarrow \mathsf{VDPF.Gen}(1^\lambda, u_i)$
2 : $l_{A,B} \leftarrow \mathsf{VDPF.Eval}(k_{A,B}, u_i)$
3 : $\overline{m} \leftarrow m \oplus \mathsf{PRG}(l_A) \oplus \mathsf{PRG}(-l_B)$
4 : $(\pi_A, \pi_B) \leftarrow \mathsf{Cloak.Proof}(x_i)$
5 : **return** $(k_A, \pi_A, \overline{m}), (k_B, \pi_B, \overline{m})$

Setup:

1. Each server $b \in \{A, B\}$ holds database $[M]_b$ of N XOR-shares.
2. Each database row is associated with a public mailbox id u_i and a public key $g^{x_i} \in \mathbb{F}_p$. Note that g is a public generator and x_i is private.

Server.Write$(k_b, \pi_b, \overline{m})$

1 : $([v]_b, \gamma_b) \leftarrow \mathsf{VDPF.Eval}\left(k_b, \{u_i\}_{i=1}^N\right)$
2 : $[v]_b \leftarrow (-1)^b \cdot [v]_b$
3 : send γ_b to other server and receive $\gamma_{b'}$
4 : **if** $\gamma_b = \gamma_{b'}$ **then**
5 : $\quad [y]_b \leftarrow \sum_{i=1}^N g^{x_i} \cdot [v_i]_b \in \mathbb{F}_p$
6 : $\quad$ **if** $\mathsf{SPOSS}([y]_b, \pi_b)$ **then**
7 : $\quad\quad$ **for** $i \in \{1, \ldots, N\}$ **do**
8 : $\quad\quad\quad \Delta \leftarrow \mathsf{PRG}([v_i]_b) \oplus (\overline{m} \cdot lsb([v_i]_b))$
9 : $\quad\quad\quad [M_i]'_b \leftarrow [M_i]_b \oplus \Delta$
10 : $\quad$ **fi**
11 : **fi**

Figure 3

Construction 1 above illustrates an improved metadata-hiding communication system in the sender-anonymous model. Setup involves two non-colluding servers with N total mailboxes. The server-side check to ensure the VDPF encodes a well-formed point function is omitted for brevity, but is an operation that incurs negligible complexity.

practical deployment. Perhaps the most glaring issues lie in the ad-hoc mechanisms of *authentication*: virtual addresses must be stored in the clear on the servers and must be significantly larger than the domain of mailboxes (to protect against guessing attacks). Although the latter primarily concerns performance, the former is far more significant: acting much like passwords, a breached address list allows unimpeded access to the set of message shares. Beyond allowing attackers to freely edit any mailbox, servers and users cannot tell which mailboxes were breached and disrupted— servers cannot tell if a client write was made by an honest or malicious user. Additionally, this solution leaves much to be desired, as it primarily leverages the implicit structure of DPFs instead of providing dedicated *authentication* methods.

Construction 1

In order to deal with the underlying limitations of Express, an alternate cryptographic protocol is presented. Preserving an identical threat model, this construction considers the problem of rejecting malicious client requests from the stance of anonymous access control and authentication.

First, we turn to an alternative method of verifying the DPF keys are well-formed, preserving the properties of the auditing protocol presented in Express. Rather than using SNIPs, this construction considers the use of VDPFs. While VDPFs by themselves are not immediately conducive to a full solution under our threat model, they are a useful first step that aggregates the client-side auditing and DPF key generation steps.

Next, using this new VDPF audit mechanism, it is possible to instantiate Cloak, a distributed Schnorr-based anonymous authentication method (S. Servan-Schreiber, personal communication, February 1,

2022). With Cloak & SPoSS, instead of storing virtual addresses, the servers hold public-key group elements: each mailbox with index i is associated with a group element g^{x_i} (S. Servan-Schreiber, personal communication, February 1, 2022). Under the intractability of the Discrete Logarithm Problem (DLP), this public-key value reveals nothing about the value of x_i (S. Servan-Schreiber, personal communication, February 1, 2022). Instead of proving knowledge of an address, clients are able to prove knowledge of the value x using a zero-knowledge proof. If a client knows a value x associated with a mailbox, they are able to write to it.

Any adversary with passive access to the list of public-keys g^x cannot access a mailbox without knowing the associated x value. Additionally, unlike the large domain size associated with virtual addressing in Express, such an approach does not require bloated mailbox indexing— that is to say, the VDPF need not expand a domain that is larger than the total number of mailboxes. The size of group $\mathbb{G}$, which in our experiments was 2048 bits, does not influence the VDPF evaluation time. This introduces notable performance benefits during both client-side key generation and each server's VDPF expansion.

One notable flaw with the above approach lies in the requirements imposed by Cloak's construction of anonymous authentication (S. Servan-Schreiber, personal communication, February 1, 2022). The VDPF in the exchange must encode *exactly* one non-zero output that is equal to 1. This is at end with the model of Express, which assumes the VDPF sent fully encodes the message m. To mitigate such incompatibility, the optimization developed by Newman et al. (2021) for arbitrary size file broadcasts can be instantiated. Their presented procedure allows for the secret sharing of an arbitrarily large XOR-share from a set of DPF keys. Using their scheme, clients send each server a random encoding of the message, along with the write request. By the properties of XOR-secret-sharing and bit-wise operations, the servers can expand the VDPF as before, while obliviously writing an arbitrarily sized message from the client.

Note that client reading within Construction 1 is established non-privately, following the model of Express. Reading within this construction must be modified due to the nature of the stored public keys. Unlike reading in Express, the address value cannot be revealed to the servers, so clients instead reveal g^x associated with their mailbox. They then use a traditional non-interactive Schnorr Proof, a zero-knowledge protocol to prove knowledge of a discrete logarithm x associated with a value g^x (Camenisch & Stadler, 1997; Schnorr, 1989).

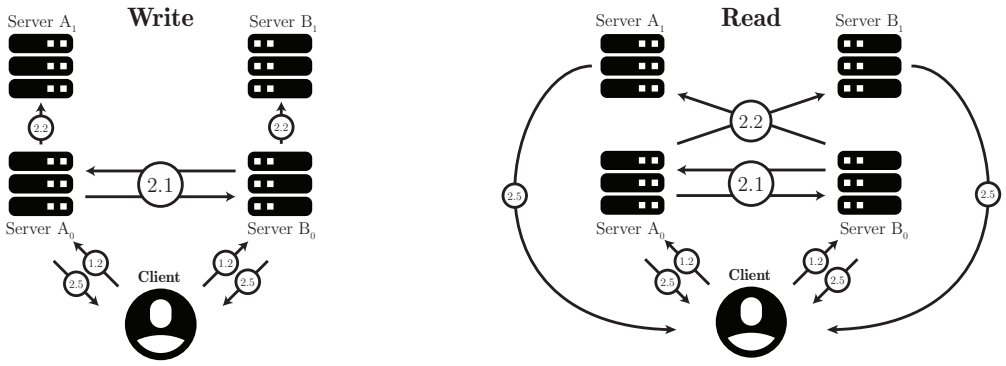
Authenticated Anonymous Reading

Achieving authenticated private reading while simultaneously preserving private writing is less trivial. In order to understand the intuition behind such a construction, it is useful to consider the primary goals of DPF-based PIR-reading.

Fundamentally, DPF-based PIR-reading works by having 2 or more parties hold *identical* databases (Boyle et al., 2016; Doerner & Shelat, 2017). When computing the inner product with the secret shared vector encoded by the DPF, each server computes a single share that is provided to the client. With PIR-writing, this setting differs in that the 2+ databases are *not identical*. In fact, each database holds different secret shares of the row contents. Reading and writing do not mesh under these models, since one requires the databases to be identical, and one requires the databases to be secret-shared.

To allow for both writing and reading, a transformation can be made to the model of Construction 1. By adding two additional servers, the two initial servers can store copies of their respective message databases (as we note in Appendix A, single-server PIR is an alternate approach to achieve this functionality in a two-server setting). Once a user wishes to read their mailbox, they authenticate and the servers perform two separate PIR-read operations. Intuitively, since each corresponding pair of servers holds identical databases, the two PIR-read operations should work as expected. For clarity, we call the original 2 servers the *authentication* servers, and the associated database-copy servers as *auxiliary* servers (e.g. authentication server A_0 corresponds to auxiliary server A_1).

Writing behavior matches that of Construction 1, but with one additional step: authentication servers send their VDPF key to their corresponding auxiliary server. This serves two purposes: (1) it shares the updated database without unreasonable communication overheads (2) it duplicates the message database in order to allow future reading. At the end of this procedure, both A servers and both B servers hold identical databases that, when combined under a bitwise-XOR operation, will evaluate to the message contents. This modified construction is reflected in Construction 2.



Authenticated Private Writing

Step 1: Form a write request.
The client does as follows:

1.1: Run $\mathsf{Client.Write}(m, u_i, x_i)$, as defined in Construction 1 ,

1.2: send $(k_A, \pi_A, \overline{m})$ to Server A_0 and $(k_B, \pi_B, \overline{m})$ to Server B_0.

Step 2: Private writing.
Upon receiving the VDPF key k_b and SPoSS proof share π_b for $b \in \{A, B\}$, servers A_0, B_0:

2.1: Run $\mathsf{Server.Write}(k_b, \pi_b, \overline{m})$, as defined in Construction 1,

2.2: if the client successfully authenticated, each server sends its key and message $(k_b, \overline{m})$ to the corresponding auxiliary servers ($A_0 \to A_1$ and $B_0 \to B_1$),

2.3: the auxiliary servers perform an identical write operation by expanding the VDPF, but do not run the authentication procedure

At this point, each server holds a message database of XOR-shares: servers A_b hold identical copies, while servers B_b also hold their own identical copy (assuming the client is honest and all servers followed protocol).

Authenticated Private Reading

Step 1: Form a read request.
The client does as follows:

1.1: Run $\mathsf{Client.Write}(m, u_i, x_i)$, as defined in Construction 1, but do not compute $\overline{m}$

1.2: send (k_A, π_A) to Server A_0 and (k_B, π_B) to Server B_0.

Step 2: Private reading.
Upon receiving the VDPF key k_b and SPoSS proof share π_b for $b \in \{A, B\}$, servers A_0, B_0:

2.1: Run $\mathsf{Server.Write}(k_b, \pi_b, \overline{m})$, as defined in Construction 1 without performing the final writing loop,

2.2: if the SPoSS proof passes, each server sends its key k_b to the opposing auxiliary servers ($A_0 \to B_1$ and $B_0 \to A_1$),

2.3: both the main and the auxiliary servers perform an identical read operation by expanding the VDPF and sending the secret share to the client. The auxiliary servers do not run the SPoSS authentication procedure

2.4: Additionally, while expanding the VDPF, each server performs an additional bitwise-XOR on each mailbox— this effectively clears the mailbox after the client has read its contents.

2.5: Each of the 4 servers sends its final, cumulative share to the client.

At this point, the client has received 4 secret shares, one from each server. In order to retrieve the mailbox contents, the client performs a bitwise-XOR on all 4 shares. Note that mailboxes are cleared in order to allow further messages to be written to the mailbox.

Figure 4
Construction 2: Anonymous Communication with Authenticated Reading & Writing.

Implementation & Evaluation

The previously presented constructions were implemented and evaluated to test their performance. All experiments were run on a Linux machine with an AMD Ryzen 5 3600 6-Core, 4.2 GHz Processor, and 32 GB RAM. Unless otherwise stated, all experiments were run in a single-threaded environment. It is emphasized that the construction presented in this work is highly parallelizable; the primary computational bottleneck, the server-side DPF expansion, can be run on each database row in parallel. For example, utilizing 8 threads would result in approximately 8× improvement in processing time. The native C/Go implementation and associated evaluations are fully open-source[4]. Additionally, as specified in RFC3526 recommended parameters[5], $\mathbb{Z}_p^*$ is instantiated with a 2048-bit safe prime.

Evaluating Costs

We compare Construction 1 to Express, another prior work that aim to provide sender-anonymous communication with cryptographic guarantees (Eskandarian et al., 2021). Express, the current state of the art, matches the setting considered in this work and serves as the best comparison to evaluate the relative successes and limitations of these constructions. It is found that Construction 1 improves client-side processing by over 100×, server-side processing by 4×, server-to-server communication costs by 8×, and overall throughput over Express. Asymptotically, both systems scale similarly, but the construction presented in this work allows for a few key improvements over Express.

Table 1 shows the asymptotic costs associated with both schemes. We note that Construction 2 enjoys the same asymptotic complexity of Construction 1, as private reading operations yield the same complexity as private writing. Practically, the costs associated with Construction 2 will be identical to Construction 1, as total latency is bounded by the speed of the authentication and the first two servers' DPF expansion time.

Client Costs

As illustrated in Figure 5, client computation incurred by this work is significantly lower than that of Express– overall processing is improved by nearly 100×, accounting for over a 99% decrease in processing time. Despite the similar asymptotic complexity (as seen in Table 1) this is achieved naturally through the methods and techniques utilized, mainly:

1. In Express, the DPF considers inputted that are at least 2^{128} (the size of a virtual address associated with a mailbox)— as has been noted, this is done by the authors to provide adequate security

[4] https://github.com/sim15/anon-com

[5] https://datatracker.ietf.org/doc/html/rfc3526

Table 1

Asymptotic and empirical costs associated with Construction 1, compared to that of Express (Eskandarian et al., 2021). Both schemes offer similar asymptotic complexity, but Construction 1 increases performance by decreasing client processing, request size, server-side processing, and inter-server data transfer. Times are reported under 10^6 user mailboxes and 1 kB messages, run on both native Go/C clients.

	Client-side			**Server-side**		
	Proof Gen.	DPF Gen.	Request Size	Audit	Write/Read	Data Transfer
Const. 1	$\mathcal{O}(1)$	$\mathcal{O}(\log n + \lvert m\rvert)$	$\mathcal{O}(\log n + \lvert m\rvert)$	$\mathcal{O}(1)$	$\mathcal{O}(n(\log n + \lvert m\rvert))$	$\mathcal{O}(1)$
		98 μs	**5.1 kB**		**2.565 s**	**1.0 kB**
Express	$\mathcal{O}(1)$	$\mathcal{O}(\log n + \lvert m\rvert)$	$\mathcal{O}(\log n + \lvert m\rvert)$	$\mathcal{O}(1)$	$\mathcal{O}(n(\log n + \lvert m\rvert))$	$\mathcal{O}(1)$
		15220 μs	5.4 kB		8.2017 s	8.3 kB

with their 128-bit statistical security parameter. Additionally, this regular DPF must encode the full message the user wishes to write. In this work, the use of a VDPF removes the need for virtual addresses, thus allowing for inputs that are only as large as the number of registered users.

2. Furthermore, the VDPF need not expand to the full message domain and can vary with the desired security of the PRG expansion algorithm (in this work, we consider 64 and 128-bit sizes).

3. The costs associated with generating the anonymous authentication proof is comparable to that of Express' SNIP auditing system, only costing a few microseconds and a constant number of field operations.

These modifications allow 2 orders of magnitude decrease in client processing using a native C/Go implementation.

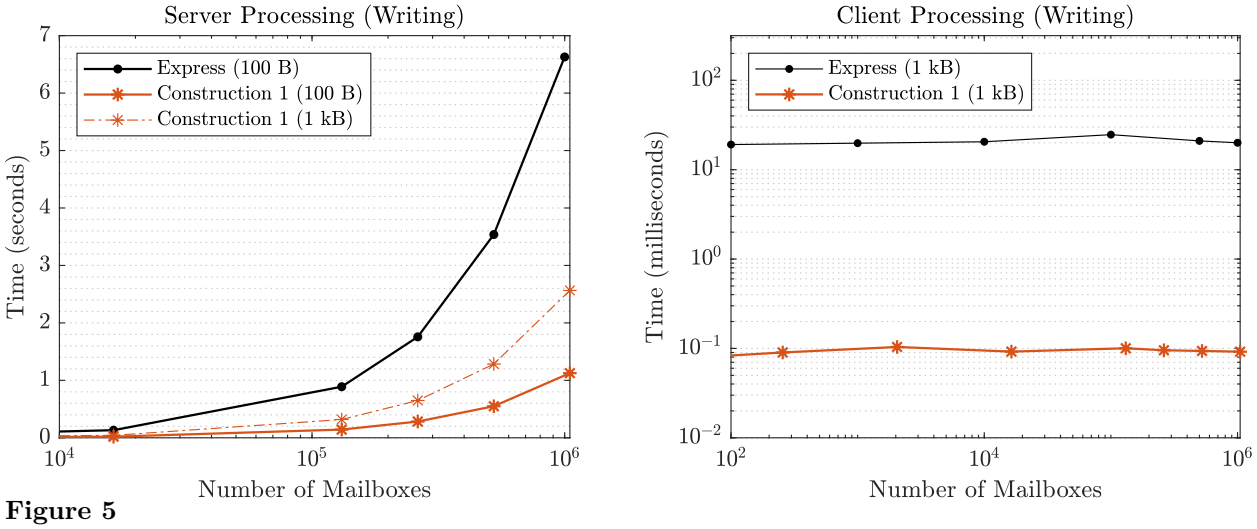


Figure 5

Server-side (A) and client-side (B) processing of the sender-anonymous construction compared to Express. Evaluated on a native Go/C implementation in a single-threaded environment. Messages exchanged between clients are set to either 100 Bytes or 1000 Bytes.

Server Costs

Server-side computation and communication, much like on the client-side, is heavily reduced. As opposed to Express, this work does not rely on SNIP-based auditing and enjoys computational speedups in part due to:

1. Decreased DPF domains decreases total computation per DPF evaluation (as described in *Client Costs*)

2. Much like Express, this work and the use of Cloak/SPoSS allows for the DPF expansion operation to be *recycled* (i.e., the DPF can be expanded once and reused for both writing and authenticating).

3. This work does not require expensive encryption/decryption operations as required by Express, though AES evaluations are still necessary during the PRG-based message expansion[6].

As shown in Figure 5, this leads to total a server-side processing time decrease of about $4 - 6\times$. Note that servers additionally do not store the user's private authentication key, unlike Express. This follows from the use of Cloak, as opposed to SNIP-based auditing. Consequently, server-to-server data transfer is further reduced by a factor of about $8\times$.

Discussion

Currently proposed systems for anonymous communication typically employ ad-hoc methods of filtering malicious clients and requests. While these systems offer strong privacy assumptions, authenticating clients in this way presents concerns in how authentication keys are stored: beyond the increased potency of data breaches, inherently high computational complexity within these schemes leaves much to be desired in terms of system efficiency. As evidenced by the above results in Figure 5 and Table 1, an improved system for metadata-hiding anonymous communication was designed and tested. Beyond illustrating a novel approach to the problem of private communication, the techniques presented in this work have succeeded in satisfying the criteria and objectives outlined previously. We comment on each of the overarching objectives in turn.

First, this work maintains the threat model and cryptographic guarantees introduced in previous work while providing solutions for both private reading and writing. This is initially achieved with a modified sender-anonymous construction that builds on the model presented in Express. Then, we illustrate one possible transformation that maintains privacy while introducing private reading. This 4-server scheme maintains similar asymptotic complexity and aligns closely to the main sender-anonymous construction.

Next, this work achieves a stronger notion of robustness against malicious clients by replacing the SNIP-based audit mechanisms of previous work with that of Cloak and SPoSS. This technique allows for the

[6] En/decryption schemes may be trivially applied to Construction 1, as in Express, to achieve further security. We expand upon this in the Discussion

Table 2
Asymptotic costs associated with anonymous communication systems with cryptographic privacy guarantees for messages of size $|m|$ and number of mailboxes/users n. The constructions presented in this work maintain a request size that is sublinear in n and constant client-server round complexity, while providing empirical performance improvements over the state of the art.

	Request Size	Rounds	Server Work	Type	Servers	Notes
Const. 1	$\mathcal{O}(\|m\| + \log n)$	1	$\mathcal{O}(n \cdot \|m\|)$	point	2	This work
Const. 2	$\mathcal{O}(\|m\| + \log n)$	1	$\mathcal{O}(n \cdot \|m\|)$	point	4	Private reading & writing
Express (2021)	$\mathcal{O}(\|m\| + \log n)$	1	$\mathcal{O}(n \cdot \|m\|)$	point	2	Primary comparison
Sabre (2021)	$\mathcal{O}(\|m\| + \log n)$	1	$\mathcal{O}(n \cdot \|m\|)$	point	2	SNIP-based auditing
Riposte (2015)	$\mathcal{O}\left(\|m\| + \sqrt{n}\right)$	1	$\mathcal{O}(n \cdot \|m\|)$	broadcast	3	Requires trusted audit server
Spectrum (2021)	$\mathcal{O}(\|m\| + \log n)$	1	$\mathcal{O}(n \cdot \|m\|)$	broadcast	2	Client cover traffic *
Blinder (2020)	$\mathcal{O}\left(\|m\| \cdot \sqrt{n}\right)$	$\log n$	$\mathcal{O}(n \cdot \|m\|)$	broadcast	5	Multi-party computation

* Cover traffic is required for many anonymous communication systems. Since many of these works depend on, and define a differing threat model, we defer discussion to these specific schemes for a more nuanced comparison.

replacement of the “virtual addressing" system of Express and ensures a passive adversary with access to user keys cannot breach the database. This aligns well with currently deployed password-based authentication systems, where user passwords cannot be stored in *the clear* on servers.

Finally, this work is shown to be practical both for clients and servers, improving client and server-side computation by around 100× and 5×, respectively. Improving client-side processing by more than 2 orders of magnitude, the techniques presented in this work represent a lightweight protocol that can be expanded to computationally weak devices or even web-based clients.

Implications & Contributions

The constructions presented in this work fall into a broad class of metadata-hiding communication systems with cryptographic protections. This work, along with many of its predecessors, specifically aims to design systems based on PIR primitives. Although this approach has been shown to be fruitful with respect to user privacy, a major point of divergence within these past systems lies in their methods of dealing with malicious behavior. This work offers an alternate approach based on anonymous access control and authentication, which is shown to fit well within the PIR model.

Table 2 illustrates the path of past literature and highlights the differences and similarities of our constructions. Much like past work utilizing PIR-based primitives, Constructions 1 and 2 have similar asymptotic costs in both request size and server-side work. Crucially, the constructions presented in this work benefit from significant performance improvements by compressing the size of the DPF keys exchanged and evaluated. This is achieved while maintaining much of the same system design as past works like

Express and Sabre (Eskandarian et al., 2021; Vadapalli et al., 2021).

Furthermore, the approach of Construction 1, requiring neither high round-complexity nor synchronous clients, suggests a path to a more abstract notion of anonymous access control within communication. Construction 2 illustrates this fact, offering full anonymity within both reading in writing at scale. The constructions presented in this work affirm that anonymous access control offered by schemes like Cloak hold potential to decrease the costs associated with existing privacy-preserving protocols.

Future Work

In addition to satisfying the aforementioned objectives, the presented constructions are broadly applicable to more complex communication flows. Future work remains to be seen in evaluating how such a point-to-point construction can be abstracted for more diverse user interactions. Private sub/pub communication (Gaballah et al., 2021), and private broadcast (Corrigan-Gibbs et al., 2015; Newman et al., 2021), are compelling examples. These works typically consider differing threat models that may benefit from the techniques presented here, both in terms of security and privacy.

Future work should also consider alternate approaches in private account selection. The private read and write construction presented in this work considers a threat model with 4 non-colluding servers, primarily due to the reduced communication complexity and privacy guarantees. Some systems may necessitate differing deployment, and thus we briefly describe three alternate design sketches.

Firstly, a two-server private read and write construction can be realized using a few differing techniques. One intuitive solution considers the use of cPIR— the single-server variant of PIR— along with the sender-anonymous model. Here, while cPIR systems come with slightly weaker privacy assumptions and certain computational restraints, the threat model can be reduced to only 2 servers. With cPIR, a separate querying process would occur following the SPoSS authentication proof. This wouldn't require separate copies of the key list on auxiliary servers.

An alternate two-server construction may consider the more non-trivial use of Oblivious RAM models (ORAM). These techniques may preserve the privacy assumptions of the models in this work but will incur high communication overheads during server-to-server data transfers (Doerner & Shelat, 2017). This may become prohibitively expensive under a large number of mailboxes but may benefit from a more robust data access model.

Conclusion

As the Internet takes a more commanding role, users increasingly look for online services to satisfy all facets of their daily lives. Online communication, perhaps the most ubiquitous example, provides one of

the fundamental goals of the Internet: to connect people. Nevertheless, the current digital economy and rise in big data have revealed the dangerous power that lies in online communication. Most notably, the amount of meaningful user data that is constantly being exchanged and leaked. More than ever, service providers and intelligence organizations have found this data to be critical, whether that be for surveillance or advertising.

This work has followed the development of a communication system that hides critical metadata exposed during interactions. This system was designed with the goals of flexibility, efficiency, and privacy. In addition to allowing for both private reading and writing, this work achieved concrete performance improvements by shifting perspective on the main challenges of past literature: primarily those of malicious client protection. By utilizing a novel zero-knowledge proof of Discrete Logarithm knowledge over secret shares, along with modified PIR techniques, we decrease client-side processing by more than two orders of magnitude, which is paired with decreased server-side processing and associated communication costs. These constructions are shown to satisfy the primary objectives and present a model that is easily extendable to both private reading and writing simultaneously.

Although the internet is built to place user privacy as a secondary consideration, it is clear that many online interactions demand more stringent protections. Nevertheless, privacy-critical applications like whistle-blowing should not be the only focus of these protections. Today, the invasive behaviors exhibited by service providers and intelligence organizations has exposed the need for consumer applications that are designed at the system level to place user privacy first. More than anything, internet users today must be cognizant of the power their data holds.

References

Abraham, I., Pinkas, B., & Yanai, A. (2020). Blinder: Mpc based scalable and robust anonymous committed broadcast. *Cryptology ePrint Archive*.

Aguilar-Melchor, C., Barrier, J., Fousse, L., & Killijian, M.-O. (2015). XPIR : Private information retrieval for everyone. *Proceedings on Privacy Enhancing Technologies*, *2016*(2), 155–174. https://doi.org/10.1515/popets-2016-0010

Bai, W., Pearson, M., Kelley, P. G., & Mazurek, M. L. (2020). Improving non-experts' understanding of end-to-end encryption: An exploratory study. *2020 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. https://doi.org/10.1109/eurospw51379.2020.00036

Beimel, A., & Ishai, Y. (2001). Information-theoretic private information retrieval: A unified construction. *International Colloquium on Automata, Languages, and Programming*, 912–926.

Beimel, A., Ishai, Y., & Kushilevitz, E. (2005). General constructions for information-theoretic private information retrieval. *Journal of Computer and System Sciences*, *71*(2), 213–247. https://doi.org/10.1016/j.jcss.2005.03.002

Berret, C. (2016). Guide to securedrop. https://core.ac.uk/download/pdf/161457023.pdf.

Bollen, J., Mao, H., & Pepe, A. (2011). Modeling public mood and emotion: Twitter sentiment and socio-economic phenomena. *Proceedings of the international AAAI Conference On Web and Social Media*, *5*(1).

Boyle, E., Gilboa, N., & Ishai, Y. (2016). Function secret sharing. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*. https://doi.org/10.1145/2976749.2978429

Camenisch, J., & Stadler, M. (1997). Proof systems for general statements about discrete logarithms. *Technical Report/ETH Zurich, Department of Computer Science*, *260*.

Chor, B., Goldreich, O., Kushilevitz, E., & Sudan, M. (1995). Private information retrieval. *Proceedings of IEEE 36th Annual Foundations of Computer Science*, 41–50. https://doi.org/10.1109/sfcs.1995.492461

Conover, M. D., Goncalves, B., Ratkiewicz, J., Flammini, A., & Menczer, F. (2011). Predicting the political alignment of twitter users. *2011 IEEE Third Int'l Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third Int'l Conference on Social Computing*. https://doi.org/10.1109/passat/socialcom.2011.34

Corrigan-Gibbs, H., Boneh, D., & Mazières, D. (2015). Riposte: An anonymous messaging system handling millions of users. *2015 IEEE Symposium on Security and Privacy*, 321–338.

de Castro, L., & Polychroniadou, A. (2021). Lightweight, maliciously secure verifiable function secret sharing. *Cryptology ePrint Archive*.

Dempsey, L., & Heery, R. (1998). Metadata: A current view of practice and issues. *Journal of Documentation*, *54*(2), 145–172. https://doi.org/10.1108/eum0000000007164

Dierks, T., & Rescorla, E. (2008). The transport layer security (tls) protocol version 1.2.

Diffie, W., & Hellman, M. (1976). New directions in cryptography. *Ideas that created the future* (pp. 421–440). The MIT Press. https://doi.org/10.7551/mitpress/12274.003.0044

Dingledine, R., Mathewson, N., & Syverson, P. (2004). *Tor: The second-generation onion router* (tech. rep.). Defense Technical Information Center. https://doi.org/10.21236/ada465464

Doerner, J., & Shelat, A. (2017). Scaling ORAM for secure computation. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. https://doi.org/10.1145/3133956.3133967

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *Advances in cryptology* (pp. 10–18). Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-39568-7_2

Endeley, R. E. (2018). End-to-end encryption in messaging services and national security—case of WhatsApp messenger. *Journal of Information Security*, *09*(01), 95–99. https://doi.org/10.4236/jis.2018.91008

Eskandarian, S., Corrigan-Gibbs, H., Zaharia, M., & Boneh, D. (2021). Express: Lowering the cost of metadata-hiding communication with cryptographic privacy. *30th USENIX Security Symposium (USENIX Security 21)*, 1775–1792.

Gaballah, S. A., Coijanovic, C., Strufe, T., & Mühlhäuser, M. (2021). 2pps—publish/subscribe with provable privacy. *2021 40th International Symposium on Reliable Distributed Systems (SRDS)*, 198–209.

Goldwasser, S. (1997). Multi party computations: Past and present. *Proceedings of the Sixteenth Annual ACM Symposium on Principles of Distributed Computing*, 1–6. https://doi.org/10.1145/259380.259405

Greenwald, G. (2014). *No place to hide: Edward snowden, the nsa, and the us surveillance state*. Macmillan.

Isaak, J., & Hanna, M. J. (2018). User data privacy: Facebook, cambridge analytica, and privacy protection. *Computer*, *51*(8), 56–59. https://doi.org/10.1109/mc.2018.3191268

Karnin, E., Greene, J., & Hellman, M. (1983). On secret sharing systems. *IEEE Transactions on Information Theory*, *29*(1), 35–41. https://doi.org/10.1109/tit.1983.1056621

Lee, J., Choi, R., Kim, S., & Kim, K. (2017). Security analysis of end-to-end encryption in telegram. *Simposio en Criptografia Seguridad Informatica, Naha, Japon. Disponible en https://bit.ly/36aX3TK*.

Mayer, J., Mutchler, P., & Mitchell, J. C. (2016). Evaluating the privacy properties of telephone metadata. *Proceedings of the National Academy of Sciences*, *113*(20), 5536–5541. https://doi.org/10.1073/pnas.1508081113

Newman, Z., Servan-Schreiber, S., & Devadas, S. (2021). Spectrum: High-bandwidth anonymous broadcast with malicious security. *IACR Cryptol. ePrint Arch.*, *2021*, 325.

Perez, B., Musolesi, M., & Stringhini, G. (2018). You are your metadata: Identification and obfuscation of social media users using metadata information. *Twelfth International AAAI Conference on Web and Social Media*.

Pohlig, S., & Hellman, M. (1978). An improved algorithm for computing logarithms over gf (p) and its cryptographic significance (corresp.) *IEEE Transactions on information Theory*, *24*(1), 106–110.

Rachels, J. (1975). Why privacy is important. *Philosophy & Public Affairs*, 323–333.

Rainie, L., & Duggan, M. (2016). Privacy and information sharing.

Rao, D., Yarowsky, D., Shreevats, A., & Gupta, M. (2010). Classifying latent user attributes in twitter. *Proceedings of the 2nd international workshop on Search and mining user-generated contents - SMUC '10*. https://doi.org/10.1145/1871985.1871993

Rescorla, E., & Dierks, T. (2018). The transport layer security (tls) protocol version 1.3. https://doi.org/10.17487/RFC8446

Riley, J. (2017). Understanding metadata. *Washington DC, United States: National Information Standards Organization*, *23*. http://www.niso.org/publications/press/UnderstandingMetadata.pdf.

Schneier, B. (2014). Metadata = surveillance. *IEEE Security & Privacy*, *12*(2), 84–84. https://doi.org/10.1109/msp.2014.28

Schnorr, C.-P. (1989). Efficient identification and signatures for smart cards. *Conference on the Theory and Application of Cryptology*, 239–252.

Shor, P. W. (1999). Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Review*, *41*(2), 303–332. https://doi.org/10.1137/s0036144598347011

Vadapalli, A., Storrier, K., & Henry, R. (2021). De l'épée au sabre for speedier ripostes.

Wang, Y. (2015). Privacy-preserving data storage in cloud using array BP-XOR codes. *IEEE Transactions on Cloud Computing*, *3*(4), 425–435. https://doi.org/10.1109/tcc.2014.2344662

Wang, Y., & Desmedt, Y. (2014). Efficient secret sharing schemes achieving optimal information rate. *2014 IEEE Information Theory Workshop (ITW 2014)*. https://doi.org/10.1109/itw.2014.6970885

Appendix A

Limitations & Assumptions

Limitations. Much like past literature, the communication flow described by this work involves point-to-point communication, where a single encoded message is to be stored on the database servers; in this work, messages are stored as XOR-shares, with one allocated to each mailbox. Subsequently, incoming write requests can only edit the *existing* mailbox contents (in this case all modifications are performed with XOR). Thus, the default model presented stands the risk of corrupting message contents if more than one message is written prior to a read request. While this means users can't immediately write more than one message at a time, potential solutions may involve allocating multiple mailbox spaces for a fixed number of messages, or increasing the size of a single mailbox according to system requirements.

Additionally, the systems presented in this work don't immediately protect against a single colluding server (i.e., a single colluding server can maliciously authenticate as a user and access a mailbox). While the SPoSS authentication protocol is susceptible to such an attack, it is worth noting that messages can still be protected using similar encryption techniques as presented in Express (Eskandarian et al., 2021).

Assumptions. The schemes in this work primarily operate under the threat models described in the *Overview*. Additionally, this work operates under the assumption that the Discrete Logarithm is computationally intractable under sufficiently large prime-order groups. This is a widely accepted assumption— known as the Discrete Logarithm Problem— in cryptography, theoretical computer science, and mathematics, as no efficient algorithm has been proposed (Shor, 1999).

Appendix B

Notation

Table B1
A set of symbols and notation utilized throughout this work. Certain symbols are omitted, as they are defined in-text during their use. Most notation typically follows the conventions of previous literature.

$[s]$	A secret share of the value s
PRG	Pseudorandom Number Generator
λ	Security parameter (statistical *or* computational)
l	Number of servers
t	Threshold for secret-sharing schemes. Indicates number of parties that must come together to reconstruct a secret.
$lsb(x)$	Least significant bit of the binary representation of some integral value x — equivalent to x mod 2.

Appendix C

Security Arguments

This appendix sketches an overview of the security properties of the two constructions presented in this work. Due to the nature of the black-box instantiations we employ throughout this work, we emphasize that these sketches are not fully formalized. While these security proofs are by no means exhaustive, our constructions benefit from adopting much of the same arguments as the primitives we rely on, and thus we aim to outline the general proof approach.

Definition 1 (Soundness). *For all probabilistic polynomial-time (PPT) adversaries $\mathcal{A}$ and for all messages $m \in \mathcal{M}$, a scheme is sound if the probability $\mathcal{A}$ successfully writes (or reads) a message to/from the database is negligible.*

Claim. Constructions 1 and 2 are sound.

Proof (sketch). The soundness of Construction 1 follows directly from the security of the utilized anonymous authentication protocol (S. Servan-Schreiber, personal communication, February 1, 2022). Since the ability of any client to interact with the database is solely dependent on the status of their authentication, we point to the security arguments of Cloak and its associated SPoSS proof. An adversary $\mathcal{A}$ is only able to successfully modify or read the database if their SPoSS authentication passes and their VDPF verification passes. The probability a client is able to create a malformed VDPF query is negligible in λ (de Castro & Polychroniadou, 2021). The soundness of Construction 2 follows immediately from the soundness of Construction 1. □

Definition 2 (Privacy/Metadata-hiding). *A sender-anonymous communication scheme is private if there exists a probabilistic polynomial-time (PPT) simulator $\mathcal{S}$ whose output is indistinguishable from the view of an adversary $\mathcal{A}$, who controls one server and an arbitrary number of corrupted clients (none of which are the designated recipients of any honest requests) (Eskandarian et al., 2021) .*

Claim. Construction 1 satisfies the sender-anonymous privacy property.

Proof (Sketch). In order to properly simulate a client write request, $\mathcal{S}$ uses the zero-knowledge simulator of the Cloak scheme and the privacy-implying simulator of the VDPF constructions: For the first part of a client request, $\mathcal{S}$ can generate DPF key under the same process of the simulator in VDPFs that implies their privacy (de Castro & Polychroniadou, 2021). The SPoSS proof sent by the client is shown to have an efficient simulator as well, which $\mathcal{S}$ can adopt analogously (S. Servan-Schreiber, personal communication, February 1, 2022). □