DATASHARENETWORK: A Decentralized Privacy-Preserving Search Engine for Investigative Journalists

Kasra Edalatnejad and Wouter Lueks, *EPFL;* Julien Pierre Martin*;* Soline Ledésert, Anne L'Hôte, and Bruno Thomas, *ICIJ;* Laurent Girod and Carmela Troncoso, *EPFL*

This paper is included in the Proceedings of the 29th USENIX Security Symposium.

August 12–14, 2020

978-1-939133-17-5

DATASHARENETWORK
A Decentralized Privacy-Preserving Search Engine for Investigative Journalists

Kasra EdalatNejad
SPRING Lab, EPFL

Wouter Lueks
SPRING Lab, EPFL

Julien Pierre Martin
Independent

Soline Ledésert
ICIJ

Anne L'Hôte
ICIJ

Bruno Thomas
ICIJ

Laurent Girod
SPRING Lab, EPFL

Carmela Troncoso
SPRING Lab, EPFL

Abstract

Investigative journalists collect large numbers of digital documents during their investigations. These documents can greatly benefit other journalists' work. However, many of these documents contain sensitive information. Hence, possessing such documents can endanger reporters, their stories, and their sources. Consequently, many documents are used only for single, local, investigations. We present DATASHARENETWORK, a decentralized and privacy-preserving search system that enables journalists worldwide to find documents via a dedicated network of peers. DATASHARENETWORK combines well-known anonymous authentication mechanisms and anonymous communication primitives, a novel asynchronous messaging system, and a novel multi-set private set intersection protocol (MS-PSI) into a *decentralized peer-to-peer private document search engine*. We prove that DATASHARENETWORK is secure; and show using a prototype implementation that it scales to thousands of users and millions of documents.

1 Introduction

Investigative journalists research topics such as corruption, crime, and corporate misbehavior. Two well-known examples of investigative projects are the Panama Papers that resulted in several politicians' resignations and sovereign states recovering hundreds of millions of dollars hidden in offshore accounts [27], and the Boston Globe investigation on child abuse that resulted in a global crisis for the Catholic Church [22]. Investigative journalists' investigations are essential for a healthy democracy [10]. They provide the public with information kept secret by governments and corporations. Thus, effectively holding these institutions accountable to society at large.

In order to obtain significant, fact-checked, and impactful results, journalists require large amounts of documents. In a globalized world, local issues are increasingly connected to global phenomena. Hence, journalists' collections can be relevant for other colleagues working on related investigations. However, documents often contain sensitive and/or confidential information and possessing them puts journalists and their sources increasingly at risk of identification, prosecution, and persecution [33, 34]. As a result journalists go to great lengths to protect both their documents and their interactions with other journalists [35]. With these risks in mind, the International Consortium of Investigative Journalists (ICIJ) approached us with this question: *Can a global community of journalists search each other's documents while minimizing the risk for them and their sources?*

Building a practical system that addresses this question entails solving five key challenges:

1) *Avoid centralizing information.* A party with access to all the documents and journalists' interaction would become a very tempting target for attacks by hackers or national agencies, and for legal cases and subpoenas by governments.

2) *Avoid reliance on powerful infrastructure.* Although ICIJ has journalists worldwide, it does not have highly available servers in different jurisdictions.

3) *Deal with asynchrony and heterogeneity.* Journalists are spread around the world. There is no guarantee that they are online at the same time, or that they have the same resources.

4) *Practical on commodity hardware.* Journalists must be able to search documents and communicate with other journalists without this affecting their day-to-day work. The system must be efficient both computationally and in communication costs.

5) *Enable data sovereignty.* Journalists are willing to share but not unconditionally. They should be able to make informed decisions on revealing documents, on a case-by-case basis.

The first four requirements preclude the use of existing advanced privacy-preserving search technologies, whereas the fifth requirement precludes the use of automatic and rule-based document retrieval. More concretely, the first requirement prevents the use of central databases and private information retrieval (PIR) [7, 23, 30] between journalists, as standard PIR requires a central list of all searchable (potentially sensitive) keywords. The second requirement rules out multi-party computation (MPC) between distributed servers [25, 40, 41].

The third and fourth requirement exclude technologies that require many round trips or high bandwidth between journalists such as custom private set intersection [13, 19, 25, 29, 42], keyword-based PIR [4, 11], and generic MPC protocols [25, 40, 41, 52], as well as the use of privacy-preserving communication systems that require all users to be online [31, 51].

We introduce DATASHARENETWORK, a decentralized document search engine for journalists to be integrated within ICIJ's open source tool for organizing information called Datashare [26]. DATASHARENETWORK addresses the challenges as follows. First, journalists keep their collections in their computers. Thus, if a journalist is hacked, coerced, or corrupted, only her collection is compromised. Second, we introduce a new multi-set private set intersection (MS-PSI) protocol that enables asynchronous search and multiplexes queries to reduce computation and communication costs. Third, we combine existing privacy-preserving technologies [3, 16] to build a pigeonhole-like communication mechanism that enables journalists to anonymously converse with each other in an unobservable manner. These components ensure that even if an adversary gains the ability to search others' documents, she cannot extract all documents nor all users in the system. In the rest of the document, for simplicity, we refer to DATASHARENETWORK as DATASHARE.

Our contributions are as follows:

✓ We elicit the security and privacy requirements of a document search system for investigative journalists.

✓ We introduce MS-PSI, a private set intersection protocol to efficiently search in multiple databases without incurring extra leakage with respect to traditional PSI with pre-computation.

✓ We propose an asynchronous messaging system that enables journalists to search and converse in a privacy-preserving way.

✓ We design DATASHARE, a secure and privacy-preserving decentralized document search system that protects from malicious users and third parties the identity of its users, the content of the queries and, to a large extent, the journalists' collections themselves. We show that DATASHARE provides the privacy properties required by journalists, and that the system can easily scale to more than 1000 participants, even if their document collections have more than 1000 documents.

2 Towards Building DATASHARE

We build DATASHARE at the request of the International Consortium of Investigative Journalists, ICIJ. When unambiguous from the context, we refer to ICIJ simply as the organization.

2.1 Requirements Gathering

In order to understand the needs of investigative journalists, ICIJ ran a survey among 70 of their members and provided us with aggregate statistics, reported below. We used the survey results as starting point for the system's requirements, and we refined these requirements in weekly meetings held for more than one year with the members of ICIJ's Data & Research Unit who are in charge of the development and deployment of the local tool Datashare [26].

User Base. ICIJ consists of roughly 250 permanent journalist members in 84 countries. These members occasionally collaborate with external reporting partners. The maximum number of reporters working simultaneously on an investigation has reached 400. The organization estimates that each member is willing to make approximately one thousand of their documents available for searching. To accommodate growth, we consider that DATASHARE needs to *scale to (at least) 1000 users, and (at least) 1 million documents*.

Journalists work and live all over the globe, ranging from Sydney to San Francisco, including Nairobi and Kathmandu; this results in large timezone differences. Around 38% of the journalists have a computer permanently connected to the Internet, and another 53% of them are connected during work hours: eight hours a day, five days a week. The rest are connected only during a few hours per day. As it is unlikely that journalists are online at the same time, *the search system needs to enable asynchronous requests and responses*. Furthermore, many journalists live in regions with low-quality networks: only half of the journalists report having a fast connection. Thus, DATASHARE *cannot require high bandwidth*.

Waiting Time. As the system must be asynchronous, the survey asked journalists how much they are willing to wait to obtain a the result of a query. 21% of the surveyees are willing to wait for hours, whereas another 56% can wait for one or more days. Hence, DATASHARE *does not need to enable real-time search*. Yet, given the delivery times of up to 24 hours, to keep search latency within a few days, DATASHARE must *use protocols that can operate with just one communication round*. Therefore, we discard multi-round techniques such as multi-party computation [25, 40, 41, 52].

Queries Nature. The queries made by journalists are in a vast majority formed by *keywords* called *named entities*: names of organizations, people, or locations of interest. Therefore, journalists do not require a very expressive querying language: DATASHARE must *support queries made of conjunctions of keywords*. Journalists are interested in a small set of these entities at a time: only those related to their current project. Consequently, *queries are not expected to include more than 10 terms at a time*, and *journalists are not expected to issue a large number of queries in parallel*.

During the design phase, we also learned that as most terms of interest are investigation-specific (e.g., XKeyScore in the Snowden leaks, or Mossack Fonseca in the Panama Papers), *a pre-defined list of terms cannot cover all potentially relevant keywords for journalists*. Therefore, techniques based on fixed lists such as private information retrieval (PIR) [7, 23, 30] are not suitable for building DATASHARE.

Security and Privacy. Regarding security and privacy concerns, journalists identify four types of principals: the journalists themselves, their sources, the people mentioned in the documents, and the ICIJ. They identify three assets: the named entities in documents, the documents themselves, and the conversations they have during an investigation. The disclosure of named entities could leak information about the investigation, or could harm the cited entities (which could in turn could trigger a lawsuit). Whole documents are considered the most sensitive as they provide context for the named entities. Finally, the disclosure of the content or existence of conversations could endanger the journalists involved, their sources, the organization, and the whole investigation.

Journalists mostly worry about third party adversaries such as corporations, governments (intelligence agencies), and organized crime. Sources and other journalists are in general considered non-adversarial. Similarly, journalists *trust ICIJ to be an authority for membership and to run their infrastructure*. However, to prevent coercion and external pressures, ICIJ does *not* want to be trusted for privacy.

The main requirement for DATASHARE is *to protect the confidentiality of assets from third parties that are not in the system*. This implies that DATASHARE cannot require journalists to send their data to third parties for analysis, storage, indexing, or search. Journalists are concerned about only subsets of these adversaries at a time. Therefore, DATASHARE *does not need to defend against global adversaries*.

Journalists initially did not consider their colleagues as adversaries. However, after a threat analysis, we concluded that there is a non-negligible risk that powerful adversaries can bribe or compromise honest journalists, in particular when those journalists live in jurisdictions with less protection for civil rights. Therefore, we require that DATASHARE *must minimize the amount of information that journalists, or ICIJ, learn about others: searched keywords, collections, and conversations*. More concretely, we require that *searches be anonymous and that the searched terms be kept confidential, with respect to both journalists and the organization*. This way neither journalists nor the organization become a profitable target for adversaries.

With respect to conversations, 64% of the surveyees report that they would prefer to remain anonymous in some cases. Furthermore, 60% of the respondents declare that they prefer to have a screening conversation before deciding to share documents. This means that *search and sharing features need to be separated to enable screening*. DATASHARE must *provide anonymous means for journalists to discuss document sharing to ensure safety*. We expect conversations within DATASHARE to be short, as their only goal is to agree on whether to proceed with sharing. After journalists agree, we assume they will switch to an alternative secure communication channel and DATASHARE does not need to support document retrieval.
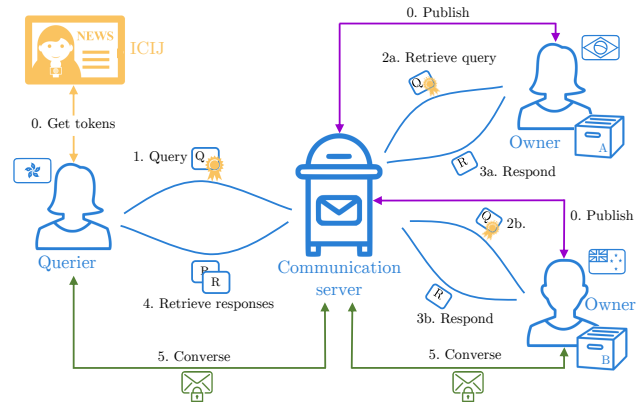

Figure 1: DATASHARE architecture overview.

2.2 Sketching DATASHARE

DATASHARE is run by ICIJ. Access to the system is exclusive to ICIJ members and authorized collaborators. Journalists trust ICIJ to act as a token issuer and only give tokens to authorized journalists. To enable journalists to remain anonymous, tokens are implemented using blind signatures. Journalists use these tokens demonstrate membership without revealing their identities.

DATASHARE provides the following infrastructure to facilitate asynchronous communication between journalists: a *bulletin board* that journalists use to broadcast information, and a *pigeonhole* for one-to-one communication. All communications between journalists and the infrastructure (pigeonhole or bulletin board) are end-to-end encrypted (i.e., from journalist to journalist) and anonymous. Hence, the infrastructure needs to be trusted for availability, but not to protect the privacy of the journalists and their documents.

Each authorized journalist in DATASHARE owns a corpus of documents that they make available for search. Journalists can take two roles: (i) *querier*, to search for documents of interest, and (ii) *document owner*, to have their corpus searched. Journalists first search for matching documents then (anonymously) converse with the corresponding document owners to request the document.

Figure 1 sketches DATASHARE's architecture. First, journalists upload privacy-preserving representations of their collections and contact information to the bulletin board. To issue a query, journalists construct a privacy-preserving representation of their keywords and broadcast it together with an authorization token through the bulletin board. Owners periodically retrieve new queries from the bulletin board. If the authorization is valid, they send a response to the querier using the pigeonhole. The querier uses this response to identify matches with the documents in the owner's collection.

When journalists find a *match* in a collection, i.e., a document that contains all the keywords in the query, they can start a conversation with the document owner to request shar-

Table 1: Notation.

$\mathbb{G}, g, p$	A cyclic group, its generator and the group's order
ℓ	The security parameter
$x \leftarrow^{\$} X$	Draw x uniformly at random from the set X
$H, \hat{H}$	Hash functions mapping into $\{0,1\}^{\ell}$ resp. group $\mathbb{G}$.
$[n]$	The set $\{1, \ldots, n\}$
s, c	The server's and client's secret keys
Y_i	The server's ith set $Y_i = \{y_{i,1}, .., y_{i,n_i}\}$
N, n_i	Nr. of server sets, resp. nr. of elements in set Y_i
X	The client's set $X = \{x_1, .., x_m\}$
m	The number of elements in the client's set
$\tau, \tau^{(i)}$	Pretags for client (τ) resp. the server's ith set Y_i ($\tau^{(i)}$)
TC	The server's tag collection

Client		Server
$X = \{x_1, .., x_m\} \subset \mathbb{G}$		$Y = \{y_1, .., y_n\} \subset \mathbb{G}$
$c \leftarrow^{\$} \mathbb{Z}_p$		$s \leftarrow^{\$} \mathbb{Z}_p$
$\tilde{x}_i = x_i^c$	$\xrightarrow{\langle \tilde{x}_i \rangle}$	$\hat{x}_i = \tilde{x}_i^s$
$\mathsf{T}_i = H(\hat{x}_i^{c^{-1}})$	$\xleftarrow{\langle \hat{x}_i \rangle, \mathsf{TC}}$	$\mathsf{TC} = \{H(y^s) \mid y \in Y\}$
Return $\{x_i \mid \mathsf{T}_i \in \mathsf{TC}\}$		

Figure 2: Vanilla PSI protocol by De Cristofaro et al. [12].

ing. Document owners append a public contact key to their collection to enable queriers to carry out this conversation in an anonymous way via the pigeonhole.

Instantiation. DATASHARE uses four main privacy-preserving building blocks: a multi-collection search mechanism, a messaging system, an anonymous communication channel, and an authorization mechanism.

We implement the privacy-preserving search mechanism by using a novel primitive that we call multi-set private set intersection (MS-PSI) described in Section 3. We design a privacy-preserving messaging system in Section 4; it provides both the bulletin board and pigeonhole functionality. We rely on the Tor [16] network as anonymous communication channel, and we use blind signatures to implement privacy-preserving authorization (see Section 5.1). In Section 5.2, we explain how DATASHARE combines these building blocks.

3 Multi-set PSI

Private set intersection (PSI) protocols enable two parties holding sets X and Y to compute the intersection $X \cap Y$, without revealing information about the individual elements in the sets. In this section, we introduce a multi-set private set intersection (MS-PSI) protocol that simultaneously computes intersections of set X with N sets $\{Y_1, \ldots, Y_N\}$ at the server. In Section 6, we review existing PSI variants.

Notation. (See Table 1) We use a cyclic group $\mathbb{G}$ of prime order p generated by g. We write $x \leftarrow^{\$} X$ to denote that x is drawn uniformly at random from the set X. Let ℓ be a security parameter. We define two hash functions $H : \{0,1\}^* \to \{0,1\}^{\ell}$ and $\hat{H} : \{0,1\}^* \to \mathbb{G}$. Finally, we write $[n]$ to denote the set $\{1, \ldots, n\}$.

Related PSI Schemes. We build on the single-set PSI protocol by De Cristofaro et al. [12], see Figure 2. In this protocol the client blinds her elements $x_i \in \mathbb{G}$ as $\tilde{x}_i = x_i^c$ using a blinding factor c before sending them to the server. The server applies its own secret to the blinded elements,

$\hat{x}_i = \tilde{x}_i^s$, and sends them back to the client in the same order, together with a tag collection of her own blinded elements: $\mathsf{TC} = \{H(y^s) \mid y \in Y\}$. The client unblinds her elements, obtaining a list of x_i^ss. Then, the client computes a tag $H(x_i^s)$ for each of them and compares it to the server's tags TC to find matching elements.

To increase efficiency when the server set is large, client-server PSI (C-PSI) schemes in the literature [19, 29, 49] introduce optimizations to avoid that the server has to compute and send a large fresh set of tags every execution. Instead, the server *precomputes* the tag collection with a long-term secret key s and sends it to the client once. In subsequent *online* phases, the server answers clients' queries by using the long-term key s. This significantly improves the communication and computation cost, as the server does not compute or send the tag collection every time.

A New Multi-set PSI Protocol. Our *multi-set* private set intersection protocol (MS-PSI) intersects a client set $X = \{x_1, .., x_m\} \subset \{0,1\}^*$ with N sets $Y_i = \{y_{i,1}, .., y_{i,n_i}\} \subset \{0,1\}^*$ at the server to obtain the intersections $X \cap Y_i$. Our protocol computes all intersections *simultaneously*, lowering the computation and communication cost with respect to running N parallel PSI protocols. In DATASHARE, X contains the query (a conjunction of search keywords) and Y_i represents document i's keywords, as described in Section 5.2. We use $\hat{H}$ to map keywords to group elements.

A naive approach to building MS-PSI would be to mimic the client-server protocols and to reuse the long-term key s for all sets Y_i. This approach maps identical elements in sets Y_i, Y_j to the same tag revealing intersection cardinalities $|Y_i \cap Y_j|$. We remove the link between tags across sets by adding a tag diversifying step to the precomputation phase of client-server PSI (see Figure 3). We first compute pretags $\tau^{(i)}$ for each set Y_i by raising each element to the power of the long-term secret s. Then, we compute per-set tags by hashing the pretags τ with the set index i to obtain $H(i \| \tau)$. The hash-function ensures that the tags of each set are independent. The server publishes the tag collection TC and the number of sets N.

During the online phase, the client blinds its set as in the scheme of De Cristofaro et al. and sends it to the server. The server re-blinds the set with its secret s and sends it back to the client in the same order. The client unblinds the result to obtain the pretags for her elements. The client then computes

Client	Server
$X = \{x_1, .., x_m\}$	$\{Y_1, .., Y_N\}$
	$Y_i = \{y_{i,1}, .., y_{i,n_i}\}$

Precomputation phase

	$s \xleftarrow{\$} \mathbb{Z}_p$
	$\tau^{(i)} = \{\hat{H}(y)^s \mid y \in Y_i\}$
$\xleftarrow{\text{TC}, N}$	$\text{TC} = \{H(i \,\|\, t) \mid$
	$\quad i \in [N] \wedge t \in \tau^{(i)}\}$

Online phase

$c \xleftarrow{\$} \mathbb{Z}_p$
$\tilde{x}_i = \hat{H}(x_i)^c \quad \xrightarrow{\langle \tilde{x}_i \rangle} \quad \hat{x}_i = \tilde{x}_i^s$
$\tau_i = \hat{x}_i^{c^{-1}} \quad \xleftarrow{\langle \hat{x}_i \rangle}$
For $d \in \{1, \ldots, N\}$:
$\quad \mathsf{T}_i^{(d)} = H(d \,\|\, \tau_i)$
Return $\{I_d = \{x_i \mid \mathsf{T}_i^{(d)} \in \mathsf{TC}\}\}_{d \in [N]}$

Figure 3: Our MS-PSI protocol.

the corresponding tags $\mathsf{T}^{(d)}$, for each document $d \in [N]$, and obtains the intersection.

In the extended version [17] (Appendix A), we prove the following theorem to show that the server learns nothing about the client's set, and that the client learns nothing more than the intersections $X \cap Y_i$.

Theorem 1. *The MS-PSI protocol is private against malicious adversaries in the random oracle model for H and $\hat{H}$, assuming the one-more-gap Diffie-Hellman assumption holds.*

The MS-PSI protocol does not provide correctness against a malicious server, who can respond arbitrarily leading the client to compute an incorrect intersection. However, from Theorem 1 we know that, even then, the malicious server cannot gain any information about the client's set.

Performance. Table 2 compares the performance of our MS-PSI protocol with the vanilla and the client-server PSI protocols in the multi-set setting. We show the computation and communication cost for a server with N sets and a client set with m elements. MS-PSI reduces the server's online communication and computation by a factor N. The client can replace expensive group operations by inexpensive hash computations, significantly reducing her online cost. The example costs for $N = 1000$ (in square brackets) illustrate this reduction showing an improvement of 3 orders of magnitude.

4 Privacy-Preserving Messaging

In this section, we introduce DATASHARE's communication system (CS). Journalists use the CS to support MS-PSI-based search and to converse anonymously after they find a match. The CS respects the organization's limitations (see

Table 2: Performance of PSI variants in a multi-set scenario: N is the number of server sets; S is the total number of server elements; m is the size of the client set; and τ_e and τ_H denote the cost of an exponentiation and a hash computation ($\tau_{H+e} = \tau_H + \tau_e$). We report in square brackets the cost estimation when $m = 10$, $N = 1000$, $S = 100,000$ (i.e., server sets have 100 elements). We assume that group elements require 32 bytes, $\tau_e = 100\,\mu s$, and $\tau_H = 1\,\mu s$.

	Vanilla	C-PSI	MS-PSI
Precomputation phase			
Server	—	$S\tau_{H+e}$	$S\tau_{H+e}$
Comms	—	S	S
Online phase			
Client	$2mN\tau_{H+e}$	$2mN\tau_{H+e}$	$2m\tau_e + mN\tau_H$
	[2 s]	[2 s]	[12 ms]
Server	$S\tau_{H+e} + mN\tau_e$	$mN\tau_e$	$m\tau_e$
	[11 s]	[1 s]	[1 ms]
Comms	$S + 2mN$	$2mN$	$2m$
	[3.84 MB]	[640 KB]	[640 B]

Section 2.1). The communication costs do not hinder the day-to-day operation of journalists, and the system supports asynchronous communication. As the organization cannot deploy non-colluding nodes, the CS uses one server. This server is trusted for availability, but not for privacy.

DATASHARE's communication system is designed to host short conversations for discussing the sharing of documents. We anticipate that journalists will migrate to using encrypted email or secure messengers if they need to communicate over a long period or if they need to send documents.

4.1 Messaging System Construction

The server provides two components: a *bulletin board* for broadcast messages, and a *pigeonhole* for point-to-point messages. We use communication server to refer to the entity that operates both components. To hide their network identifiers from the server and network observers, journalists always use Tor [16] for communication. To ensure unlinkability, DATASHARE creates a new Tor circuit for every request.

Bulletin Board. The bulletin board implements a database that stores broadcast messages. Journalists interact with the bulletin board by using two protocols: BB.broadcast(m), which adds a message m to the database to broadcasts it to all journalists, and $m \leftarrow$ BB.read() to retrieve unseen messages.

Pigeonhole. The pigeonhole consists of a large number of one-time-use mailboxes. Journalists use the pigeonhole to send and receive replies to search queries and to conversation messages. Journalists use the method PH.SendRaw (Protocol 1) to send query replies; and the asynchronous process PH.RecvProcess (Protocol 2) to retrieve incoming query

replies and conversation messages. Journalists use PH.Monitor (Protocol 3) to receive notifications of new messages from the pigeonhole and to trigger PH.RecvProcess. Journalists are expected to connect to the system several times a week (see Section 2.1). In agreement with ICIJ, we decided that the pigeonhole will delete messages older than 7 days.

Journalists may initiate a conversation after receiving a successful match. To hide this event, we ensure that the *sending of conversation messages is unobservable*: the server cannot determine whether a journalist sends a conversation message or not (see Definition 1). This hides whether a conversation occurred, and therefore whether the search revealed a match or not. To ensure unobservability of conversation messages, journalists run PH.Cover (Protocol 4) to send cover messages at a constant Poisson rate to *every* journalist. To send a conversation message, it suffices to replace one of the cover messages with the real message (see PH.HiddenSend, Protocol 5).

Journalists use the Diffie-Hellman key exchange to compute mailbox addresses and message encryption keys, and an authenticated encryption scheme AE to encrypt messages. Queriers generate a fresh key for every query and use that key to receive query replies and to send conversation messages associated with that query. Document owners use a medium-term key to send query replies and to receive conversation messages from queriers (see Section 5.2). When exchanging cover traffic, journalists use fresh cover keys to send and their medium-term keys to receive.

Protocol 1 (PH.SendRaw($\mathsf{sk}_S, \mathsf{pk}_R, m$))**.** To send message m to recipient R with public key pk_R, a sender with private key sk_S proceeds as follows. Let n_s be the number of times S called PH.SendRaw to send a message to R before. The sender
  1. computes the Diffie-Hellman key $k' = \mathsf{DH}(\mathsf{sk}_S, \mathsf{pk}_R)$;
  2. computes the random rendezvous mailbox $\mathsf{addr} = H(\text{'addr'} \,||\, k' \,||\, \mathsf{pk}_S \,||\, n_s)$ and a symmetric key $k = H(\text{'key'} \,||\, k' \,||\, \mathsf{pk}_S \,||\, n_s)$;
  3. pads the message m to obtain m' of length mlen, and computes the ciphertext $c = \mathsf{AE.enc}(k, m')$;
  4. opens an anonymous connection to the pigeonhole and uploads c to mailbox addr.
For every upload, the pigeonhole notifies all monitoring receivers (see PH.Monitor below) that a message arrived at addr.

Protocol 2 (PH.RecvProcess($\mathsf{sk}_R, \mathsf{pk}_S$))**.** To receive a message from sender S with public key pk_S, a receiver R with private key sk_R runs the following asynchronous process. Let n_r be the number of times R successfully received a message from S. The receiver
  1. computes the Diffie-Hellman key $k' = \mathsf{DH}(\mathsf{sk}_R, \mathsf{pk}_S)$;
  2. uses k' to compute a random rendezvous mailbox $\mathsf{addr} = H(\text{'addr'} \,||\, k' \,||\, \mathsf{pk}_S \,||\, n_r)$ and a symmetric key $k = H(\text{'key'} \,||\, k' \,||\, \mathsf{pk}_S \,||\, n_r)$;
  3. waits until PH.Monitor (see below) receives a notification of a new message on address addr. If no message is posted to addr in seven days, the process terminates;

4. opens an anonymous connection to the pigeonhole and downloads the ciphertext c at address addr (if there was no message due to a false positive, the process continues at step 3); and
5. decrypts the message $m' = \mathsf{AE.dec}(k, c)$ and returns the unpadded message m or $\perp$ if decryption failed.
When the receiver goes offline, this process is paused and resumed when the receiver comes online again.

A sender may send multiple messages without receiving a response. The receiver calls PH.RecvProcess repeatedly to receive all messages (n_r increases every time). To ensure that the participants derive the correct addresses and decryption keys, participants keep track of the message counters n_s, n_r for each pair of keys $(\mathsf{sk}_S, \mathsf{pk}_R)$ and $(\mathsf{sk}_R, \mathsf{pk}_S)$, respectively.

Protocol 3 (PH.Monitor)**.** Journalists run the PH.Monitor process to monitor for incoming messages. The receiver
  1. opens an anonymous monitoring connection to the pigeonhole and requests a list of addresses addr that received a message since she was last online
  2. via the same anonymous connection, receives notifications of addresses addr with new messages.
Addresses addr received in step 1 or 2 can cause the PH.RecvProcess processes to continue past step 3. To save bandwidth, the pigeonhole sends a cuckoo filter [20] that contains the addresses in step 1. Moreover, the pigeonhole only sends the first two bytes of the address in step 2 (PH.RecvProcess handles false positives).

The PH.Cover and PH.HiddenSend protocols ensure conversation messages are unobservable. Senders store a queue of outgoing conversation messages for each recipient.

Protocol 4 (PH.Cover(sk_R))**.** As soon as the journalists come online, they start the PH.Cover process. Let sk_R be the medium-term private key, and $\mathsf{pk}_1, \ldots, \mathsf{pk}_{n-1}$ be the medium-term public keys of the other journalists. The process runs the following concurrently:
  - *Cover keys.* Draw an exponential delay $t_k \leftarrow \mathsf{Exp}(1/\lambda_k)$, and wait for time t_k. Generate a fresh cover key-pair $(\mathsf{sk}_c, \mathsf{pk}_c)$ and upload pk_c to the bulletin board by calling $\mathsf{BB.broadcast}(\mathsf{pk}_c)$. Repeat.
  - *Sending messages.* Wait until the first cover key has been uploaded. For each recipient pk_i, proceed as follows:
    1. Draw $t_i \leftarrow \mathsf{Exp}(1/\lambda_c)$ and wait for time t_i.
    2. If the send queue for pk_i is not empty, let m_i be the first message in the queue and sk_q the corresponding query key. Send the message by calling PH.SendRaw($\mathsf{sk}_q, \mathsf{pk}_i, m_i$) and remove m_i from the queue. Otherwise, let sk_c be the most recent private cover key and m_i be a dummy message. Send the message by calling PH.SendRaw($\mathsf{sk}_c, \mathsf{pk}_i, m_i$).
    3. Repeat.
  - *Receiving cover messages.* For each of the non-expired cover keys pk_c' on the bulletin board, call the process

$m \leftarrow$ PH.RecvProcess($\mathsf{sk}_R, \mathsf{pk}'_c$). If m is a real message (see Section 5.2) forward the message to DATASHARE, otherwise discard. Repeat.

This process stops when the user goes offline, and PH.RecvProcess processes started by PH.Cover are canceled.

Protocol 5 (PH.HiddenSend($\mathsf{sk}_S, \mathsf{pk}_R, m$)). To send a message m to recipient R with public key pk_R, sender S with private key sk_S places m in the send queue for pk_R.

4.2 Messaging Service Privacy

We first define unobservability then prove that conversation messages sent using PH.HiddenSend are unobservable.

Definition 1 (Unobservability). A conversation message is unobservable if all PPT adversaries have a negligible advantage in distinguishing a scenario in which the sender S sends a conversation message to the receiver R, from a scenario where S does not send a conversation message to R.

Theorem 2. *Messages sent using PH.HiddenSend are unobservable towards any adversary that controls the communication server but does not control the sender or the receiver, assuming the receiver awaits both conversation and cover messages. This statement is also true when the adversary can break the network anonymity Tor provides.*

Proof. To show that conversation messages are unobservable, we must prove that the following two scenarios are indistinguishable: the scenario in which the sender sends a conversation message (sent by PH.Cover after a conversation message has been queued using PH.HiddenSend), and the scenario in which the sender sends a cover message (sent by PH.Cover when no conversation message has been queued). The intuition behind this proof is that the conversation and cover messages are indistinguishable: (1) both are encrypted so that the adversary cannot distinguish them based on content; and (2) conversation messages replace cover messages, so they are sent using the same schedule.

All messages go through the pigeonhole. For each message, the adversary observes (1) the pigeonhole address, (2) the content, (3) the length, (4) the timestamps at which the message was posted and retrieved, and – in the worst case scenario in which the adversary can break the anonymity Tor provides – (5) the sender and the receiver.

The content and pigeonhole address of messages are cryptographically indistinguishable. Senders and receivers compute rendezvous mailbox addresses by using a Diffie-Hellman key exchange based on either the query public key and the owner's public key (when the message is a conversation messages) or the sender and receiver's cover keys (when the message is a cover message). As the adversary does not control the sender or the receiver, it does not know the corresponding private keys in either scenario. Under the decisional Diffie-Hellman

assumption, the adversary cannot distinguish between mailbox addresses for conversation messages and mailbox addresses for cover messages. Under the same DH assumption, the adversary cannot learn the symmetric key k that is used to encrypt the message either. Moreover, all messages are padded to a fixed length of mlen. Hence, the adversary cannot distinguish between the two situations based on message content or length. As a result, all messages sent between sender S and receiver R are indistinguishable to the adversary on the cryptographic layer.

We now show that the post and retrieve times of the messages are also independent of whether the message is a cover message or a conversation message:

Sender. The "cover keys" and "sending messages" processes of PH.Cover are, by design, independent of whether a conversation message should be sent or not. The sender sends (real or cover) messages to the recipient at a constant rate λ_c. The send times are independent of whether the sender has a real message for the receiver.

Receiver. The receiver is listening to both conversation and cover messages from the sender. As soon as it a new message notification arrives, PH.RecvProcess will retrieve this message. Therefore, the retrieval time does *not* depend on the type of message. □

As a corollary of the unobservability proof, we have the following theorem.

Theorem 3. *The pigeonhole protects the secrecy of messages from non-participants including the communication server.*

To hide their (network) identities from the communication server, users of DATASHARE communicate with the communication server via Tor. Sender anonymity hides queriers' identities from document owners, and receiver anonymity hides document owners' identities from queriers. Using Tor ensures these properties, even when journalists collude with the communication server. Formally, we define sender and receiver anonymity as follows:

Definition 2 (Sender anonymity). A communication system provides sender anonymity if any PPT adversary has a negligible advantage in guessing the sender of a message.

Definition 3 (Receiver anonymity). A communication system provides receiver anonymity if any PPT adversary has a negligible advantage in guessing the receiver of a message.

Theorem 4. *Assuming that Tor provides sender and receiver anonymity with respect to the communication server, the communication system provides sender and receiver anonymity at the network layer against adversaries who control the communication server and a subset of journalists.*

Proof. All messages go through the communication system and journalists never directly connect with each other. We
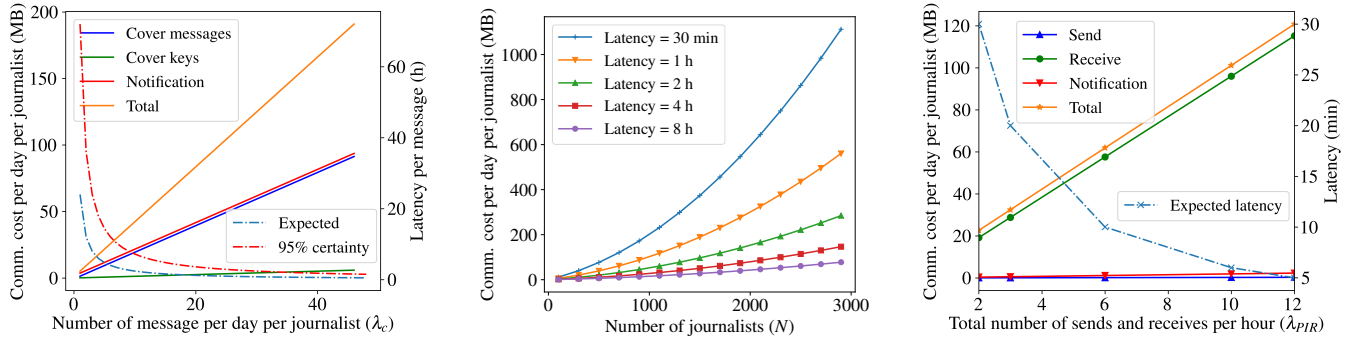
Figure 4: Left: bandwidth (left axis) and latency (right axis) for running the communication system (CS) with 1000 journalist for given rate λ_c. Middle: varying the number of journalists and average latency in the CS. Right: bandwidth (left axis) and latency (right axis) for running the PIR system with 1000 journalists.

study separately the anonymity provided by the bulletin board and the pigeonhole.

To publish an encrypted message (the query) to the bulletin board, senders run the BB.broadcast protocol over a fresh Tor circuit. Sender anonymity is guaranteed by Tor. The bulletin board broadcasts all messages to all journalists. As these messages do not have an intended receiver, receiver anonymity is not relevant.

Both senders and receivers use fresh Tor circuits when communicating with the communication servers. This ensures that communications are unlinkable at the network layer, and that the adversary cannot identify the journalist from network artifacts. As shown in the unobservability proof, the pigeonhole cannot distinguish senders' or receivers' given addresses or encrypted messages. ☐

This theorem only addresses the anonymity at the network layer. We discuss anonymity at the application layer, i.e., based on the content of messages, in Section 5.3.

Tor does not provide sender or receiver anonymity against global passive adversaries. To protect against global passive adversaries, DATASHARE will migrate to stronger network layer anonymity systems (e.g., the Nym system [47], based on Loopix [44])

4.3 Cost Evaluation

To guarantee unobservability, we schedule the traffic according to a Poisson distribution. However, such strong protection comes at a cost [15]: Regardless of whether they have zero, one, or many conversations, every journalist sends messages at a rate λ_c to the other N journalists, i.e., sends $\lambda_c N$ messages per day. Consequently, every journalist also receives $\lambda_c N$ messages a day.

Figure 4, left, illustrates the trade-off between bandwidth overhead and latency for a given cover traffic rate. When journalists send few messages a day, the bandwidth requirements are very low. For instance, setting λ_c to be 4 messages per day

requires every journalist to use 16.5 MB per day, including the sending of notifications and the updating of cover keys. For these messages to be unobservable, however, journalists have to wait on average six hours between messages (less than 18 hours in 95% of the cases). If journalists require higher throughput they must consume more bandwidth. For example, setting $\lambda_c = 48$ messages a day ensures that messages are sent within half an hour on average (and within 90 minutes with probability 95%). Storing messages from the last seven days on the pigeonhole for 1000 journalists and send rate of $\lambda_c = 48$ requires 390 GB, which is manageable for a server.

The latency we report in Figure 4 assumes that journalists are online. If they disconnect from the system before a message is sent, journalists must, after coming online again, first upload a new cover key then draw a new sample from $\text{Exp}(\lambda_c)$ to decide when to send their message. We propose to set the update latency λ_k to $\lambda_c/4$, so that the initial latency is at most 25% more than the latency under normal circumstances.

For the current size of the population that will use DATASHARE, 250 journalists (see Section 2.1), the bandwidth can be kept reasonable at the cost of latency. However, as journalists send cover traffic to everyone, the bandwidth cost increases quadratically with the size of the population, and becomes pretty heavy after reaching 2000 journalists, see Figure 4, center.

An Alternative Construction. If the traffic requirements become too heavy for the organization members, bandwidth can be reduced by increasing the computation cost at the pigeonhole server. Instead of using cover traffic to *all* journalists to hide the mailboxes that contain real messages, journalists can retrieve messages using computational private information retrieval (PIR) [3, 30].

In this approach, senders send cover messages at a rate λ_{PIR}, *independent* of the number of journalists, to random mailboxes. When they have a real message, they send it instead of a cover message. They use the same rate to retrieve messages using PIR. This approach hides which messages

are getting retrieved from the pigeonhole and breaks the link between the send and receive time. As a result, the server's observation of the system is independent of whether journalists send a real message or not.

We illustrate the trade-off associated with this approach in Figure 4, right. We use SealPIR [3] to retrieve cover and conversation messages. Responding to a PIR request in a scenario of 1000 journalists and a send rate of 6 messages per hour takes 12 seconds. Therefore, we assume a server with 24 cores (approx 1300 USD/month in AWS) can handle this scenario. We see that this approach enables the system to send conversation messages at a higher rate and a lower cost. For example, sending 6 messages per hour (144 messages a day) requires around 59 MB. However, as opposed to the Poisson cover approach described in the previous section, this rate limits the total number of messages per day *regardless of recipient*. As a result, depending on the number of receivers journalists want to communicate with on average, one or the other method could be more advantageous.

5 The DATASHARE System

We now present DATASHARE, an asynchronous decentralized peer-to-peer document search engine. DATASHARE combines the multi-set private set intersection protocol (Section 3), the privacy-preserving communication system (Section 4), and an anonymous authentication mechanism.

5.1 Preliminaries

Processing Documents. The primary interests of investigative journalists are named entities, such as people, locations, and organizations (see Section 2.1). ICIJ has already developed a tool [26] that uses natural language processing to extract named entities from documents. After the extraction, the tool transforms named entities into a canonical form to reduce the impact of spelling variation in names. We employ this tool to canonicalize queries. An advantage of using this tool over simply listing all words in a document is that it reduces the number of keywords per document: the majority of documents have less than 100 named entities.

Search. DATASHARE uses the MS-PSI protocol as a pairwise search primitive between journalists. The querier acts as MS-PSI client, and the client's set represents the querier's search keywords. The document owners act as MS-PSI servers, where the server's N sets represent the keywords in each of the owner's N documents. Each document owner has their own *different* corpus and secret key. We say a document is a match if it contains *all* query keywords (i.e., the conjunction of the query keywords, see Section 2.1). MS-PSI speeds up the computation and reduces the communication cost by a factor of N compared to the naive approach of running one PSI protocol per document.

Authenticating Journalists. Only authorized journalists, such as members of the organization or collaborators, are allowed to make queries and send conversation messages. DATASHARE's authentication mechanism operates in epochs. In each epoch journalists obtain a limited number of anonymous tokens. Tokens can be used only once, which limits the number of queries that journalists can make per epoch. Compromised journalists, therefore, can extract limited information from the system by making search queries. We considered using identity-escrow mechanisms to mitigate damage by misbehaving journalists but in agreement with the organization, we decided against this approach as such mechanisms could too easily be abused to identify honest journalists.

Recall from Section 2.1 that journalists trust the organization as an authority for membership and already have means to authenticate themselves to the organization. Therefore, the organization is the natural design choice for issuing anonymous tokens. We note that, even if the organization is compromised, it can do limited damage as it cannot link queries or conversations to journalists (because of token anonymity). However, it can ignore the rate limit. This would enable malicious queriers to extract more information than allowed. To mitigate this risk, DATASHARE could also work with several token issuers and require a threshold of valid tokens.

For the epoch duration, ICIJ proposes one month to provide a good balance between protection and ease of key management. Rate-limits are flexible. The organization can decide to provide additional one-time-use tokens to journalists who can motivate their need for extra tokens. Although this reveals to the organization which journalists are more active, it does not reveal what they use the tokens for.

Instantiation. Tokens take the form of a blind signature on an ephemeral signing key. We use Abe's blind signature (BS) scheme [1]. The organization runs $\mathsf{BS.Setup}(1^\ell)$ to generate a signing key msk and a public verification key mpk. To sign an ephemeral key pk_T, the journalist and the organization jointly run the $\mathsf{BS.Sign}()$ protocol. The user takes as private input the key pk_T, and the organization takes as private input its signing key msk. The user obtains a signature C on pk_T. The verification algorithm $\mathsf{BS.Verify}(\mathsf{mpk}, C, \mathsf{pk}_T)$ returns $\top$ if C is a valid for pk_T and $\bot$ otherwise. These blind signatures are anonymous. The blindness property of BS ensures that the signer cannot link the signature C or the key pk_T to the journalist that ran the corresponding signing protocol.

Let sk_T be the private key corresponding to pk_T. We call the tuple $T = (\mathsf{sk}_T, C)$ an authentication token. Journalists use tokens to authenticate themselves before issuing a query or sending a message. To authenticate themselves, journalists create a signature σ on the message using sk_T and append the signature σ and blind signature C on pk_T. Non-authenticated messages and queries are dropped by other journalists.

Anonymous authentication with rate limiting could have been instantiated alternatively with *n*-times anonymous credentials [9], single show anonymous credentials [6, 8], or

regular anonymous credentials [5, 45] made single-show. We opted for the simplest approach.

Cuckoo Filter. DATASHARE uses cuckoo filters [20] to represent tag collections in a space-efficient manner. The space efficiency comes at the price of having false positives when answering membership queries. The false negative ratio is always zero. The false positive ratio is a parameter chosen when instantiating the filter. Depending on the configuration, a cuckoo filter can compress a set to less than two bytes per element regardless of the elements' original size.

Users call CF.compress(S, params) to compute a cuckoo filter CF of the input set S using the parameters specified in params. Then, CF.membership(CF, x) returns $\top$ if x was added to the cuckoo filter, and $\bot$ otherwise. For convenience, we write CF.intersection(CF, S') to compute the intersection $S' \cap S$ with the elements S contained in the cuckoo filter. The function CF.intersection can be implemented by running CF.membership on each element of S'.

5.2 DATASHARE Protocols and Design

The journalists' organization sets up the DATASHARE system by running SystemSetup (Protocol 6). Thereafter, journalists join DATASHARE by running JournalistSetup (Protocol 7). Journalists periodically call GetToken (Protocol 8) to get new authentication tokens, and Publish (Protocol 9) to make their documents searchable. DATASHARE does not support multiple devices, and the software running on journalists' machines automatically handles key management without requiring human interaction. If a journalist's key is compromised, she contacts the organization to revoke it.

Protocol 6 (SystemSetup). The journalist organization runs SystemSetup to set up the DATASHARE system:

1. The organization generates a cyclic group $\mathbb{G}$ of prime order p with generator g, and hash functions $H : \{0,1\}^* \to \{0,1\}^\ell$ and $\hat{H} : \{0,1\}^* \to \mathbb{G}$ for use in the MS-PSI protocol. It selects parameters params for the cuckoo filter and sets the maximum number of query keywords lim (we use lim = 10). The organization publishes these.
2. The organization sets up a token issuer by running (msk, mpk) = BS.Setup(1^ℓ) and publishes mpk.
3. The organization sets up a communication server, which provides a bulletin board and a pigeonhole.

Protocol 7 (JournalistSetup). Journalists run JournalistSetup to join the network: The journalist authenticates to the organization and registers for DATASHARE.

Protocol 8 (GetToken). Journalists run GetToken to obtain one-time-use authentication tokens from the organization.

1. The journalist J connects to the organization and authenticates herself. The organization verifies that J is allowed to obtain an extra token and, if not, aborts.

2. The journalist generates an ephemeral signing key (sk_T, pk_T); runs the BS.Sign() protocol with the organization to obtain the organization's signature C on the message pk_T (without the organization learning pk_T); and stores the token $T = (\mathsf{sk}_T, C)$.

To obtain tokens for the new epoch, journalists repeatedly run the GetToken protocol at the beginning of each epoch.

Protocol 9 (Publish). Journalists run Publish to make their documents searchable. Publish takes as input a token $T = (\mathsf{sk}_T, C)$ and a set Docs = $\{d_1, .., d_N\}$ of N documents such that each document d_i is a set of keywords in $\{0,1\}^*$. This protocol includes the pre-computation phase of MS-PSI.

1. The journalist chooses a secret key $s \xleftarrow{\$} \mathbb{Z}_p$ and computes her tag collection for the MS-PSI protocol as

$$\mathsf{TC} = \{H(i \,||\, \hat{H}(y)^s) \mid i \in [N], \, y \in d_i\},$$

and compresses it into a cuckoo filter CF = CF.compress(TC, params).
2. The journalist generates a long-term pseudonym nym, and a medium-term contact key pair (sk, pk).
3. The journalist encodes her pseudonym nym, public key pk, compressed tag collection CF, and the number of documents N as her public record

$$\mathsf{Rec} = (\mathsf{nym}, \mathsf{pk}, \mathsf{CF}, N).$$

4. The journalist signs her record $\sigma = \mathsf{Sign}(\mathsf{sk}_T, \mathsf{Rec})$ and runs BB.broadcast(Rec $||\, \sigma \,||\, \mathsf{pk}_T \,||\, C$) to publish it.

DATASHARE automatically rotates (e.g., every week) the medium-term contact key of journalists (sk, pk) to ensure forward secrecy. This prevents that an attacker that obtains a journalist's medium-term private key can recompute the mailbox addresses and encryption key of messages sent and received by the compromised journalist.

Journalists retrieve all public records from the bulletin board. They run Verify(pk_T, σ, Rec) to verify the records against the ephemeral signing key, check that they have not seen pk_T before to enforce the one-time use, and run BS.Verify(pk_T, C, mpk) to validate the blind signature. Journalists discard invalid records.

DATASHARE incorporates MS-PSI into its protocols to enable document search. Querying works as follows (Fig. 5): (1) The querier posts a query together with a fresh key pk_q to the bulletin board (Protocol 10); (2) Document owners retrieve these queries from the bulletin board (2a), they compute the reply address, and they send the reply to a pigeonhole mailbox (2b, see Protocol 11); (3) The querier monitors the reply addresses for all document owners, retrieves the replies, and computes the intersection to determine matches (Protocol 12).

Protocol 10 (Query). Queriers run Query to search for keywords X. The protocol takes as input a token $T = (\mathsf{sk}_T, C)$.
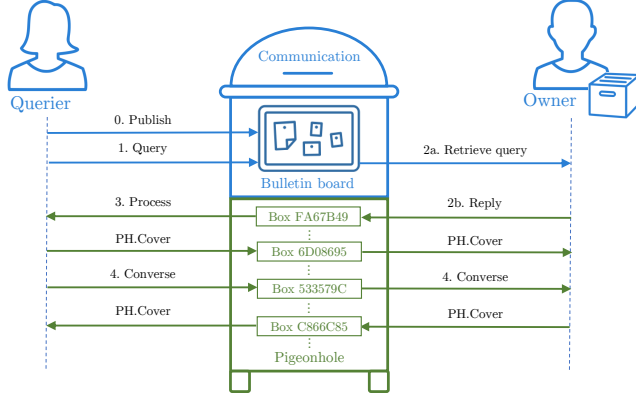
Figure 5: An overview of DATASHARE protocols.

1. The querier generates a key pair $(\mathsf{sk}_q, \mathsf{pk}_q)$ for the query and pads X to lim keywords by adding random elements.
2. As in the MS-PSI protocol, the querier picks a fresh blinding factor $c \xleftarrow{\$} \mathbb{Z}_p$, and computes:

$$Q = \{\hat{H}(x)^c \mid x \in X\}.$$

3. The querier signs the query Q and her public key pk_q as $\sigma = \mathsf{Sign}(\mathsf{sk}_T, Q \,\|\, \mathsf{pk}_q)$, and broadcasts the query Q, public key pk_q, signature σ, ephemeral token key pk_T, and token C by running $\mathsf{BB.broadcast}(Q \,\|\, \mathsf{pk}_q \,\|\, \sigma \,\|\, \mathsf{pk}_T \,\|\, C)$.

Recall that MS-PSI perfectly hides the keywords inside queries. As a result, these queries can be safely broadcasted.

Protocol 11 (Reply). Document owners run Reply to answer a query $(Q, \mathsf{pk}_q, \sigma, \mathsf{pk}_T, C)$ retrieved from the bulletin board.
1. The owner verifies the query by checking $\mathsf{Verify}(\mathsf{pk}_T, \sigma, Q \,\|\, \mathsf{pk}_q)$, $\mathsf{BS.Verify}(\mathsf{mpk}, C, \mathsf{pk}_q)$, and that she did not see pk_T before. If any verification fails, she aborts.
2. The owner uses her secret key s to compute the MS-PSI response $R = \{\tilde{x}^s \mid \tilde{x} \in Q\}$ to the query.
3. Let sk be the owner's medium-term private key. She runs $\mathsf{PH.SendRaw}(\mathsf{sk}, \mathsf{pk}_q, R)$ to post the result to the pigeonhole, and starts the process $\mathsf{PH.RecvProcess}(\mathsf{sk}, \mathsf{pk}_q)$ to await conversation messages from the querier (see Converse below).

Protocol 12 (Process). Queriers run the Process protocol for every journalist J with record $\mathsf{Rec} = (\mathsf{nym}, \mathsf{pk}, \mathsf{CF}, N)$ to retrieve and process responses to their query (X, sk_q, c), where X is the unpadded set of query keywords.
1. The querier runs the asynchronous protocol $R \leftarrow \mathsf{PH.RecvProcess}(\mathsf{sk}_q, \mathsf{pk})$ to get the new response.
2. Similar to MS-PSI, the querier computes the size of the intersection I_i for each document d_i, $1 \le i \le N$, as

$$I_i = \left| \mathsf{CF.intersection}\left(\mathsf{CF}, \{H(i \,\|\, \hat{x}^{c^{-1}}) \mid \hat{x} \in R\}\right) \right|.$$

3. Let $q = |X|$ be the number of query keywords. The querier learns that the owner nym has $t = |\{i \mid I_i = q\}|$ matching documents.

After finding a match, the querier and owner can converse via the pigeonhole to discuss the sharing of documents using the Converse protocol.

Protocol 13 (Converse). Let $(\mathsf{sk}_q, \mathsf{pk}_q)$ be the query's key pair, and $(\mathsf{sk}_O, \mathsf{pk}_O)$ the owner's medium-term key pair at the time of sending the query.
- The querier sends messages m to the owner by calling $\mathsf{PH.HiddenSend}(\mathsf{sk}_q, \mathsf{pk}_O, m)$, and awaits replies by calling $\mathsf{PH.RecvProcess}(\mathsf{sk}_q, \mathsf{pk}_O)$.
- The owner sends messages m to the querier by calling $\mathsf{PH.HiddenSend}(\mathsf{sk}_O, \mathsf{pk}_q, m)$, and awaits replies by calling $\mathsf{PH.RecvProcess}(\mathsf{sk}_O, \mathsf{pk}_q)$.
- After receiving a message, the receiving party calls $\mathsf{PH.RecvProcess}$ again, to await further messages.

Both the query's key pk_q and the owner's key pk_O are signed using a one-time-use token. Thus, querier and owner know they communicate with legitimate journalists.

5.3 DATASHARE Security Analysis

DATASHARE provides the following guarantees:

Protecting Queries. The requirements established in Section 2.1 state that DATASHARE must protect the searched keywords and identity of the querier from adversaries that control the communication server and a subset of document owners. The Query protocol, which handles sending queries, is based on MS-PSI. It represents searched keywords as the client's set in MS-PSI. Theorem 1 states that MS-PSI perfectly hides the client's set from malicious servers. Therefore, DATASHARE protects the content of queries from owners.

DATASHARE does not reveal any information about the identity of queriers at the network and application layer. Theorem 4 ensures that the communication system provides sender and receiver anonymity and protects the querier's identity at the network layer. At the application layer, the querier sends $(Q \,\|\, \mathsf{pk}_q \,\|\, \sigma \,\|\, \mathsf{pk}_T \,\|\, C)$ as part of the Query protocol to the bulletin board. The values σ, pk_T, and C form an anonymous authentication token based on Abe's blind signature [1]. Anonymous tokens are independent of the querier's identity. The value pk_q is an ephemeral public key, and Q is a MS-PSI query which uses an ephemeral secret for the client. Hence, both pk_q and Q are independent of the querier's identity too. Therefore, the content of the query does not leak the querier's identity at the application layer.

Protecting Conversations. According to the requirements stated in Section 2.1, DATASHARE must protect (1) the content, and (2) the identity of participants in a conversation from non-participants. (3) DATASHARE must protect the identities of journalists (who are in a conversation) from each other.

First, DATASHARE protects the content of conversation messages from non-participants: Theorem 3 proves that only the sender and receiver can read their conversation messages.

Second, DATASHARE protects the identity of participants in a conversation from non-participants. Theorem 2 proves that communication is unobservable, as long as participants are awaiting both conversation and cover messages. DATASHARE enforces the conditions by construction. Immediately after answering a query (see Reply, Protocol 11), the owner starts PH.RecvProcess to listen for messages from the querier. Similarly, the querier starts to listen for conversation messages from the owner right after sending him a conversation message (see Converse, Protocol 13). Moreover, the "cover keys" and "receiving cover messages" processes in the PH.Cover protocol ensure that all journalists broadcast their cover keys and start PH.RecvProcess after receiving a new cover key. Therefore, DATASHARE satisfies the requirements on the communication systems in Theorem 2. As a result, non-participants cannot detect whether users communicate. Thus, protecting the identity of participants as required.

Third, DATASHARE aims to hide the identity of journalists from their counterparts in a conversation. Theorem 4 shows that the communication system does not reveal the identity of journalists at the network layer. DATASHARE also ensures protection at the cryptographic layer: as we argued above, queries are unlinkable. However, DATASHARE cannot provide unconditional protection for conversations. Queriers or document owners could identify themselves as part of the conversation. Moreover, by their very nature, messages in a conversation are linkable. Also, as we discuss below, insiders can use extra information to identify communication partners.

Protecting Document Collections. *Any* functional search system inherently reveals information about the documents that it makes available for search: To be useful it must return at least one bit of information. An attacker can learn more information by making additional queries. We show that DATASHARE provides comparable document owner's privacy to that of ideal theoretical search systems. We use as a security metric the number of queries an attacker has to make to achieve each of the following goals:

Document Recovery. Given a target set of keywords (e.g. "XKeyscore" and "Snowden"), an adversary aims to learn which of these target keywords are contained in a document for which some keywords are already known.

Corpus Extraction. Given a set of target keywords, an adversary aims to learn which documents in a corpus contain which target keywords. If the target set contains all possible keywords, the adversary effectively recovers the full corpus.

Any functional search system is also susceptible to confirmation attacks. An adversary interested in knowing whether a document in a collection contains a keyword (e.g., "XKeyscore" to learn whether the collection contains the Snowden documents) can always directly query for the keyword of interest.

We compare the number of queries an adversary needs to extract the corpus or recover a document in the following three settings: when using DATASHARE, and when using one of two

Table 3: Privacy and scalability of the hypothetical and DATASHARE's MS-PSI based search protocols. The table shows the number of queries necessary to achieve document recovery and corpus extraction, when interacting with a corpus of d documents over a set n keywords. The document extraction bound for the 1-bit system extracts up to uniqueness bound u.

	Doc	Extract	Scale
1-bit	n	$n^u + nd$	--
#doc	n	nd	-
DATASHARE	**n/lim**	**n/lim**	**+**

hypothetical systems. The first hypothetical system, called *1-bit*, is an ideal search system. In this system, given a query, the querier learns *only* one bit of information: whether the owner has a matching document. The second hypothetical system, called *#doc*, is an ideal search system where the querier learns how many matching documents the owner has.

Table 3 compares these hypothetical systems with DATASHARE's use of MS-PSI, where d is the number of documents and n the number of relevant keywords. We show that extracting all the keywords from a document requires at most n queries in the 1-bit and #docs search systems in the extended version [17] (Appendices B.1 and B.2).

Extracting the full corpus using the 1-bit search system is not always possible. Let the *uniqueness number* u_D be the smallest number of keywords that uniquely identify a document D. If D is a strict subset of another document D', the document cannot be uniquely identified, and we set $u_D = \infty$. However, as corpora are small, we expect that most documents can be identified by a few well-chosen keywords, resulting in small uniqueness numbers.

In Appendix B.1 of the extended version [17], we show that extracting all documents with uniqueness number less or equal to u takes $O(n^u + nd)$ queries in the 1-bit search system. In Appendix B.2 of the extended version [17], we show that extracting all documents (regardless of uniqueness number) takes $O(nd)$ queries in the #doc search system.

In DATASHARE, we limit MS-PSI queries to lim keywords per query. Hence, any document extraction attack must make at least n/lim queries to ensure all keywords are queried at least once. In fact, this bound is tight for both document recovery and corpus extraction for MS-PSI: By making n/lim queries with lim keywords each, the attacker learns which keywords are contained in which documents.

In summary, DATASHARE offers similar protection against corpus extraction as the #doc ideal system. For document recovery, not even the ideal 1-bit-search system offers much better protection. At the same time, MS-PSI is much more efficient than their ideal counterparts.

Internal Adversaries. We now discuss how an adversary may use auxiliary information about a journalist's behavior or corpus to gain an advantage in identifying the journalist. Some of these attacks are inherent to all systems that provide search or messaging capabilities. These attacks, however, *do not permit* the adversary to extract additional information from journalists' corpora.

Intersection Attacks. A malicious sender (respectively, receiver) who has access to the online/offline status of journalists can use this information to reduce the anonymity set of the receiver (respectively, sender) to only those users that are online. As more messages are exchanged, this anonymity set becomes unavoidably smaller [28]. This attack is inherent to all low-delay asynchronous messaging systems, including the one provided by the communication server. In the context of DATASHARE, we note that once document owners and queriers are having a conversation, it is likely that they reveal their identity to each other. Yet, we stress that preserving anonymity and, in general, that minimizing the digital traces left by the journalists in the system is very important to reducing the risk that journalists become profitable targets for subpoenas or hacking attempts.

Stylometry. A malicious receiver can use stylometry, i.e., linguistic style, to guess the identity of the sender of a message. The effectiveness of this attack depends on the volume of conversation [32, 37]. This attack is inherent to all messaging systems, as revealing the content of the messages is required to provide utility.

Partial Knowledge of Corpus. Adversaries who have prior knowledge about a journalist's corpus can use this knowledge to identify this journalist in the system. However, due to MS-PSI's privacy property (see Theorem 1), learning *more* about the documents in this journalist's corpus requires making search queries.

In particular, if an adversary convinces a journalist to add a document with a unique keyword pattern to his corpus, then the adversary can detect this journalist's corpus by searching for the pattern. DATASHARE cannot prevent such out-of-band watermarking. However, the adversary still needs to make further queries to learn anything about non-watermarked documents in the collection.

Non-goals. Finally, we discuss security properties that are not required in DATASHARE.

Query Unlinkability. DATASHARE does not necessarily hide which queries are made by the same querier. Even though anonymity is ensured at the network and application layers, queriers that have made multiple queries may retrieve responses for all these queries in quick succession after coming online. Document owners know the corresponding query of their messages, and if they collude with the communication server, then they can infer that the same person made these queries. As no adversary can learn any information about the queries themselves, we consider this leakage to be irrelevant.

Owner Unlinkability. DATASHARE also reveals *which*

pseudonymous document owner created a MS-PSI response, making responses linkable. DATASHARE cannot provide unlinkability for document owners when using MS-PSI. Although MS-PSI itself could be modified to work without knowing the document owner's pseudonym, an adversary could simply repeat a specific rare keyword (for example, "one-word-to-link-them-all") and identify the document owners based on the corresponding pretag that they produce for the rare keyword. We believe that revealing the document owner's pseudonym is an acceptable leakage for the performance gain it provides.

5.4 Cost Evaluation

At the time of writing, ICIJ has implemented the local search and indexing component of DATASHARE [26]. In addition, we have implemented a Python prototype of the cryptographic building blocks underlying search (Section 3) and authentication (Section 5.1).[1] We did not implement the messaging service (Section 4), as it relies on standard building blocks and cryptographic operations.

To agree on the final configuration of the system, we are currently running a user study among the organization members. The goal is to familiarize journalists with a type of search and messaging system that is different than those they typically use in their daily activities (Google and email or instant messaging, respectively), as well as with the threat model within which DATASHARE provides protection. We recall that DATASHARE hides all key management and cryptography from the users, hence we do not study those aspects.

In this section, we evaluate the performance of the cryptographic operations involved in search and authentication. Our prototype uses the petlib [14] binding to OpenSSL on the fast NIST P-256 curve for the elliptic curve cryptography in MS-PSI. We implement the Cuckoo filter using cuckoopy [2]. We ran all experiments on an Intel i3-8100 processor running at 3.60GHz using a single core. We note that operations could be easily parallelized to improve performance.

We focus our evaluation on the computational cost and bandwidth cost of the authentication and search primitives to ensure that DATASHARE fulfills the requirements in Section 2.1 without journalists needing fast hardware or fast connections. When reporting bandwidth cost, we omit the overhead of the meta-protocol that carries messages between system parties. We do not consider any one-time setup cost or the standard cryptography used for messaging. We also do not measure network delay as the latency the Tor network introduces – around one second [48] – is negligible compared to the waiting time imposed by connection asynchrony; and it is orders of magnitude less than the journalists waiting limits (see Section 2.1).

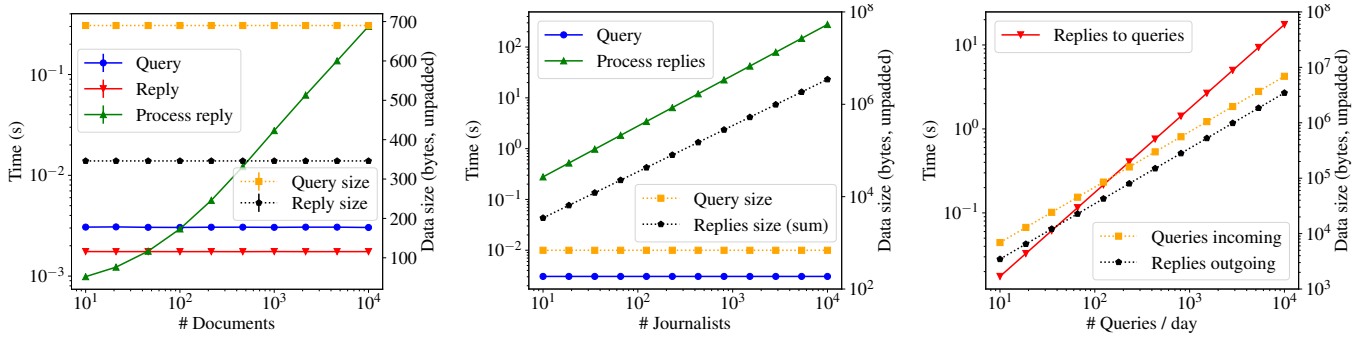[1]The code is open source and available at: https://github.com/spring-epfl/datashare-network-crypto

Figure 6: Time (left axis) and bandwidth (right axis, unpadded) for single query on one journalist (left), single query on all journalists (center), answering several queries (right).

We provide performance measurements for different system work loads. We consider the *base scenario* to be 1000 journalists, each of whom makes 1000 documents available for search. There is no requirement for the number of keywords per document or keywords per query. For a *conservative* estimate, we assume that each document contains 100 keywords, and that each query contains 10 keywords.

Authenticating Journalists. We implement the BS scheme using Abe's blind signatures [1]. Running BS.Sign requires transferring 413 bytes and takes 0.32 ms and 0.62 ms, respectively, for the organization and the journalist. Each blind signature is 360 bytes, and verifying it using BS.Verify takes 0.4 ms. We include these costs in the respective protocols.

Publishing Documents. Data owners run Publish to make their documents searchable. For the base scenario, this *one-time* operation takes 14 seconds and results in a cuckoo filter of size 400 KB for a FPR of 0.004%. For a conservative estimation, we assume all keywords are different. When documents contain duplicate elements y, the precomputation can be amortized: the pretag $\hat{H}(y)^s$ has to be computed only once.

Querying a Single Journalist. Figure 6, left, shows the time and bandwidth required to issue one query on one collection, depending on the collection size. The querier constructs the query using Query and sends it to the document owner (the querier's computation cost includes the cost of obtaining the one-time-use token using GetToken). The document owner responds using Reply. These operations are independent of the number of documents. The querier runs Process to retrieve the responses, and to compute the intersection of query and collection. This takes 27 ms in the base scenario. Bandwidth cost reflects the raw content size. But recall that, in practice, the messaging system pads messages to 1 KB.

Querying All Journalists. As expected, the processing time and bandwidth of Query are independent of the population size, whereas the cost of processing the responses grows linearly with the number of queried journalists (Figure 6, center). For the baseline scenario, processing all 999 responses takes about 27 seconds *in total* and requires retrieving 1 MB of
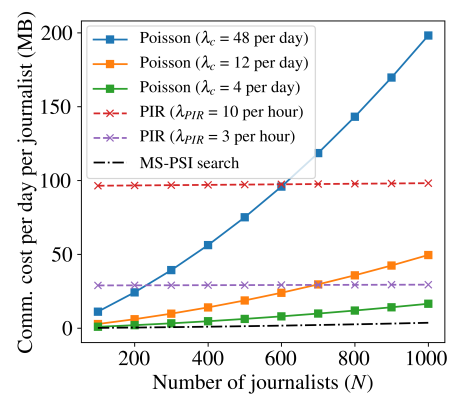

Figure 7: Communication cost for different communication strategies, depending on the number of journalists. We assume 1 query per journalist per day in the search component.

padded responses. We note that this cost is only paid by the querier, and does not impact the document owners (see below). Moreover, as replies are unlikely to arrive all at once, processing can be spread out over time; thus reducing the burden on the querier's machine.

This computation assumes that each journalist has the same number of documents. In practice, this might not hold. However, as we see in Figure 6, left, as soon as collections have more than 50 documents the computation time grows linearly with the collection size. Hence, as long as journalists have collections with at least 50 documents, the measurements in Figure 6, center, are largely independent of how these documents are distributed among journalists.

The Cost for Document Owners. Document owners spend time and bandwidth to answer queries from other journalists. Figure 6, right, shows how these costs depend on the total number of queries an owner receives per day. Even when all journalists make 10 queries of 10 keywords each day (unlikely in practice) the *total* computation time for document owners is less than 20 seconds; and they send and receive less than 7 megabytes (10 MB when padded).

Overall Cost of DATASHARE. Finally, we plot in Figure 7 the *total* bandwidth a journalist needs per day to run DATASHARE, depending on the number of journalists in the system and the strategy implemented by the communication system. Regardless of the size of the system, the cost associated to hide communications dominates the cost stemming from searches. Regarding the communication cost, as explained in Section 4.3, for small organizations Poisson-rate cover traffic provides a better trade-off with respect to throughput, but as more journalists join the system, the PIR-based system starts performing better.

6 Related Work

Many PSI protocols [13, 24, 29, 38] differ from that of De Cristofaro et al. [12], but only in how they instantiate the oblivious pseudorandom functions (OPRFs). Our MS-PSI protocols can easily be adjusted to use alternative OPRFs to compute the pretags. As bandwidth is at a premium in our scenario, we base our MS-PSI protocols on the scheme of De Cristofaro et al. as it has the lowest communication cost.

The restrictions on computational power and bandwidth rule out many other PSI schemes. Protocols based on oblivious polynomial evaluation [21] have very high computational cost. Hash-based PSI protocols [41–43] have low computational cost, but require much communication. Finally, PSI protocols can be built from generic secure multi-party computation directly [25, 40, 41]. However, this approach also suffers from a high communication cost and requires more than one communication round.

Secure multi-party computation based PSI protocols can be extended to provide better privacy than MS-PSI: The underlying circuits can be extended to implement either the ideal 1-bit search or the #doc search system. However, their high communication and round complexity rule out their use in our document search system. Recently, Zhao and Chow proposed a threshold PSI protocol based on polynomial evaluation [53] that can implement the #doc search system (by setting the threshold equal to the number of keywords). But its communication and computation complexity rule it out.

A document search engine could also be implemented using private information retrieval (PIR): Queriers use PIR to privately query keywords in the document owner's database. Computational PIR protocols [3, 30, 36] (IT-PIR protocols [7, 23] do not apply) place a high computational burden on the database owner. More importantly, PIR requires a fixed set of keywords, that cannot exist for the journalists' use case. Keyword-based PIR approaches [4, 11] sidestep this issue, but instead require multiple communication rounds. Therefore, PIR cannot be used in our scenario.

Encrypted databases hide the queries of data owners from an untrusted database server [18, 39, 46, 50]. Although DATASHARE could operate such a central encrypted database, this system would not be secure. On the one hand, if the en-

crypted database is used as a central service for all collections, then a collusion between a journalist and the database server would leak the entire database. This would violate document privacy. On the other hand, if each journalist operates a personal database, then collusion between the database server and the document owner (acting as the 'data owner' in the terminology used in the encrypted database literature) might leak search queries, as these systems are not designed to hide queries from a database server that colludes with the data owner. This would violate query privacy.

7 Future Steps: Better Protection

We have introduced DATASHARE, a decentralized privacy-preserving search engine that enables journalists to find and request information held by their peers. DATASHARE has great potential to help journalists collaborate in uncovering cross-border crimes, corruption, or abuse of power.

Our collaboration with a large organization of investigative journalists (ICIJ) provided us with a novel set of requirements that, despite being deeply grounded in practicality, are rarely considered in academic publications. These requirements led us to design new building blocks that we optimized for security trade-offs different than previous work. We combined these building blocks into an efficient and low-risk decentralized search system.

Yet, DATASHARE's protections are not perfect. Both the search primitive, and the availability of timestamps of actions in the system, leak information. At the time of writing, the high cost in bandwidth and/or computation of state-of-the-art techniques that could prevent this leakage – e.g., PIR to hide access patterns and efficient garbled circuits to implement one-bit search – precludes their deployment.

We hope that this paper fosters new research that addresses these problems. We believe that the new set of requirements opens an interesting new design space with much potential to produce results that have a high impact, not only by helping investigative journalism to support democratic societies, but also in other domains.

References

[1] Masayuki Abe. A secure three-move blind signature scheme for polynomially many signatures. In *EUROCRYPT*, 2001.

[2] Rajath Agasthya. cuckoopy: Pure python implementation of cuckoo filter. https://github.com/rajathagasthya/cuckoopy. Accessed: June 22, 2020.

[3] Sebastian Angel, Hao Chen, Kim Laine, and Srinath T. V. Setty. PIR with Compressed Queries and Amortized Query Processing. In *S&P*, 2018.

[4] Sebastian Angel and Srinath T. V. Setty. Unobservable Communication over Fully Untrusted Infrastructure. In *OSDI*. USENIX Association, 2016.

[5] Man Ho Au, Willy Susilo, Yi Mu, and Sherman S. M. Chow. Constant-size dynamic *k*-times anonymous authentication. *IEEE Systems Journal*, 2013.

[6] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In *CCS*, 2013.

[7] Amos Beimel and Yuval Ishai. Information-Theoretic Private Information Retrieval: A Unified Construction. In *ICALP*, 2001.

[8] Stefan A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy*. MIT Press, Cambridge, MA, USA, 2000.

[9] Jan Camenisch, Susan Hohenberger, Markulf Kohlweiss, Anna Lysyanskaya, and Mira Meyerovich. How to win the clonewars: efficient periodic *n*-times anonymous authentication. In *CCS*, 2006.

[10] Andrea Louise Carson. *Investigative journalism, the public sphere and democracy: the watchdog role of Australian broadsheets in the digital age*. PhD thesis, University of Melbourne, 2013.

[11] Benny Chor, Niv Gilboa, and Moni Naor. Private Information Retrieval by Keywords. Technical Report TR CS0917, Department of Computer Science, Technion, Israel, 1997.

[12] Emiliano De Cristofaro, Paolo Gasti, and Gene Tsudik. Fast and Private Computation of Cardinality of Set Intersection and Union. In *CANS*, 2012.

[13] Emiliano De Cristofaro and Gene Tsudik. Practical Private Set Intersection Protocols with Linear Complexity. In *FC*, 2010.

[14] George Danezis. Petlib: A python library that implements a number of privacy enhancing technolgies. https://github.com/gdanezis/petlib. Accessed: June 22, 2020.

[15] Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, and Aniket Kate. Anonymity Trilemma: Strong Anonymity, Low Bandwidth Overhead, Low Latency - Choose Two. In *S&P*, 2018.

[16] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The Second-Generation Onion Router. In *USENIX Security Symposium*, 2004.

[17] Kasra Edalatnejad, Wouter Lueks, Julien Pierre Martin, Soline Ledésert, Anne L'Hôte, Bruno Thomas, Laurent Girod, and Carmela Troncoso. DatashareNetwork: A Decentralized Privacy-Preserving Search Engine for Investigative Journalists. *CoRR*, abs/2005.14645, 2020. https://arxiv.org/abs/2005.14645.

[18] Mohammad Etemad, Alptekin Küpçü, Charalampos Papamanthou, and David Evans. Efficient Dynamic Searchable Encryption with Forward Privacy. *PoPETs*, 2018.

[19] Brett Hemenway Falk, Daniel Noble, and Rafail Ostrovsky. Private Set Intersection with Linear Communication from General Assumptions. *IACR Cryptology ePrint Archive*, 2018.

[20] Bin Fan, David G. Andersen, and Michael Kaminsky. Cuckoo Filter: Better Than Bloom. *;login:*, 2013.

[21] Michael J. Freedman, Kobbi Nissim, and Benny Pinkas. Efficient private matching and set intersection. In *EUROCRYPT*, 2004.

[22] Boston Globe. Church allowed abuse by priest for years. https://www.bostonglobe.com/news/special-reports/2002/01/06/church-allowed-abuse-priest-for-years/cSHfGkTIrAT25qKGvBuDNM/story.html, 2002. Accessed: June 22, 2020.

[23] Ian Goldberg. Improving the Robustness of Private Information Retrieval. In *S&P*, 2007.

[24] Carmit Hazay and Yehuda Lindell. Efficient Protocols for Set Intersection and Pattern Matching with Security Against Malicious and Covert Adversaries. *J. Cryptology*, 2010.

[25] Yan Huang, David Evans, and Jonathan Katz. Private Set Intersection: Are Garbled Circuits Better than Custom Protocols? In *NDSS*, 2012.

[26] ICIJ. Datashare. https://datashare.icij.org/. Accessed: June 22, 2020.

[27] ICIJ. Panama papers. https://www.icij.org/investigations/panama-papers/. Accessed: June 22, 2020.

[28] Dogan Kesdogan, Dakshi Agrawal, and Stefan Penz. Limits of anonymity in open environments. In *Information Hiding*, 2002.

[29] Ágnes Kiss, Jian Liu, Thomas Schneider, N. Asokan, and Benny Pinkas. Private Set Intersection for Unequal Set Sizes with Mobile Applications. *PoPETs*, 2017.

[30] Eyal Kushilevitz and Rafail Ostrovsky. Replication is NOT needed: SINGLE database, computationally-private information retrieval. In *FOCS*, 1997.

[31] David Lazar, Yossi Gilad, and Nickolai Zeldovich. Karaoke: Distributed private messaging immune to passive traffic analysis. In *OSDI*, 2018.

[32] Fernanda López-Escobedo, Carlos-Francisco Méndez-Cruz, Gerardo Sierra, and Julián Solórzano-Soto. Analysis of stylometric variables in long and short texts. *Procedia-Social and Behavioral Sciences*, 2013.

[33] Susan E. McGregor, Polina Charters, Tobin Holliday, and Franziska Roesner. Investigating the computer security practices and needs of journalists. In *USENIX*, 2015.

[34] Susan E. McGregor, Franziska Roesner, and Kelly Caine. Individual versus Organizational Computer Security and Privacy Concerns in Journalism. *PoPETs*, 2016.

[35] Susan E. McGregor, Elizabeth Anne Watkins, Mahdi Nasrullah Al-Ameen, Kelly Caine, and Franziska Roesner. When the Weakest Link is Strong: Secure Collaboration in the Case of the Panama Papers. In *USENIX*, 2017.

[36] Carlos Aguilar Melchor, Joris Barrier, Laurent Fousse, and Marc-Olivier Killijian. XPIR : Private Information Retrieval for Everyone. *PoPETs*, 2016.

[37] G MuthuSelvi, GS Mahalakshmi, and S Sendhilkumar. Author attribution using stylometry for multi-author scientific publications. *Advances in Natural and Applied Sciences*, 2016.

[38] Moni Naor and Omer Reingold. Number-theoretic constructions of efficient pseudo-random functions. *J. ACM*, 2004.

[39] Vasilis Pappas, Fernando Krell, Binh Vo, Vladimir Kolesnikov, Tal Malkin, Seung Geol Choi, Wesley George, Angelos D. Keromytis, and Steven M. Bellovin. Blind seer: A scalable private DBMS. In *S&P*, 2014.

[40] Benny Pinkas, Thomas Schneider, Gil Segev, and Michael Zohner. Phasing: Private Set Intersection Using Permutation-based Hashing. In *USENIX*, 2015.

[41] Benny Pinkas, Thomas Schneider, Christian Weinert, and Udi Wieder. Efficient Circuit-Based PSI via Cuckoo Hashing. In *EUROCRYPT*, 2018.

[42] Benny Pinkas, Thomas Schneider, and Michael Zohner. Faster Private Set Intersection Based on OT Extension. In *USENIX*, 2014.

[43] Benny Pinkas, Thomas Schneider, and Michael Zohner. Scalable private set intersection based on OT extension. *ACM Trans. Priv. Secur.*, 2018.

[44] Ania M. Piotrowska, Jamie Hayes, Tariq Elahi, Sebastian Meiser, and George Danezis. The Loopix Anonymity System. In *USENIX*, 2017.

[45] David Pointcheval and Olivier Sanders. Short Randomizable Signatures. In *CT-RSA*, 2016.

[46] Raluca A. Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. CryptDB: protecting confidentiality with encrypted query processing. In *SOSP*. ACM, 2011.

[47] Nym project. The nym system. `https://nymtech.net/`. Accessed: June 22, 2020.

[48] Tor project. Tor metrics - performance. `https://metrics.torproject.org/onionperf-buildtimes.html`. Accessed: June 22, 2020.

[49] Amanda C Davi Resende and Diego F Aranha. Faster unbalanced private set intersection. *FC*, 2018.

[50] Dawn Xiaodong Song, David A. Wagner, and Adrian Perrig. Practical Techniques for Searches on Encrypted Data. In *S&P*, 2000.

[51] Jelle van den Hooff, David Lazar, Matei Zaharia, and Nickolai Zeldovich. Vuvuzela: scalable private messaging resistant to traffic analysis. In *SOSP*, 2015.

[52] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Authenticated Garbling and Efficient Maliciously Secure Two-Party Computation. In *CCS*, 2017.

[53] Yongjun Zhao and Sherman S. M. Chow. Can You Find The One for Me? In *WPES*, 2018.