Do You Feel a Chill? Using PIR against Chilling Effects for Censorship-resistant Publishing*

Miti Mazmudar
University of Waterloo
Waterloo, ON, Canada
miti.mazmudar@uwaterloo.ca

Stan Gurtler
University of Waterloo
Waterloo, ON, Canada
tmgurtler@uwaterloo.ca

Ian Goldberg
University of Waterloo
Waterloo, ON, Canada
iang@uwaterloo.ca

ABSTRACT

Peer-to-peer distributed hash tables (DHTs) rely on volunteers to contribute their computational resources, such as disk space and bandwidth. In order to incentivize these node operators of privacy-preserving DHTs, it is important to prevent exposing them to the data that is stored on the DHT and/or queried for. Vasserman et al.'s CROPS aimed at providing plausible deniability to server nodes by encrypting stored content. However, node operators are still exposed to the contents of queries. We provide an architecture that uses information-theoretic private information retrieval to efficiently render a server node incapable of determining what content was retrieved in a given request by a user. We simulate our system and show that it has a small communication and performance overhead over other systems without this privacy guarantee, and significantly smaller overheads than the closest related work.

CCS CONCEPTS

• **Social and professional topics** → **Technology and censorship**; • **Security and privacy** → **Privacy-preserving protocols**.

KEYWORDS

Censorship-resistant publishing; query privacy; private information retrieval

ACM Reference Format:
Miti Mazmudar, Stan Gurtler, and Ian Goldberg. 2021. Do You Feel a Chill? Using PIR against Chilling Effects for Censorship-resistant Publishing. In *Proceedings of the 20th Workshop on Privacy in the Electronic Society (WPES '21), November 15, 2021, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3463676.3485612

1 INTRODUCTION

Censorship-resistant systems allow users access to online content when direct access to such content is restricted by a nation-state adversary, namely a *censor*. For instance, Tor [5] is an anonymity network that supports users in accessing websites that are censored by nation-states. Censorship-resistant *publishing* (CRP) systems allow publishers to submit their work to multiple servers, or *nodes*, such that a censor cannot take down or tamper with the published content. Several censorship-resistant publishing systems have been proposed, such as Vasserman et al.'s CROPS [14], Waldman and Mazières' Tangler [16] and Stubblefield and Wallach's Dagster [13].

CRP systems are commonly built atop structured peer-to-peer (P2P) networks, as they support redundancy and do not suffer from a single point of failure. Structured P2P networks link peer nodes' identifiers to the content that they store and use distributed hash tables (DHTs) for routing search and insertion queries.

Node operators within the censor's region of influence may be legally obliged to report any prohibited content that is stored on their machines, or that is requested by users in queries. Many node operators would thus rather not learn the content that they are storing or the content of the queries that they are receiving. The aforementioned systems support node operators in *plausibly denying* any knowledge of the content that they store, by encrypting chunks of publishers' documents and storing key material separately from these chunks. Within Vasserman et al.'s CROPS, these chunks are indexed by hashes of keywords that describe the document. However, as the keywords' hashes are exposed in clients' queries, these node operators can no longer plausibly deny knowing what content was queried.

We provide plausible deniability over queried content, by integrating Private Information Retrieval (PIR) for DHTs. We begin with a description of the building blocks we use, namely PIR, DHTs, and CRP systems, in Section 2. We discuss robust DHTs, including related work in query privacy over DHTs, in Section 3, and describe our threat model in Section 4. In Section 5 we provide an architecture allowing users to retrieve documents from the network without revealing what document was retrieved to the target node that stores the document, nor to any in-path nodes. We simulate the latency and throughput overheads of our system, to demonstrate its viability for deployment in Section 6.

2 BACKGROUND

In this section, we describe the building blocks used in our system, namely private information retrieval (PIR), distributed hash tables (DHTs) and attacks on them, as well as censorship-resistant publishing systems (CRPS) with a focus on plausible deniability.

Private Information Retrieval (PIR). PIR allows public databases, held by one or more untrusted servers, to be queried by clients while guaranteeing that the servers cannot learn which record was accessed. Information-theoretic PIR (IT-PIR) schemes require multiple servers, each of which have copies of the database [4] and they do not require any computational hardness assumptions. In IT-PIR schemes, the client constructs one query per server and using the

*An extended version of this paper is available [10].

WPES '21, November 15, 2021, Virtual Event, Republic of Korea

ACM ISBN 978-1-4503-8527-5/21/11…$15.00
https://doi.org/10.1145/3463676.3485612

responses to these queries, it can reconstruct the desired record of the database. A certain threshold number of these servers must not share the queries they receive, or else they could reconstruct the desired record. This is known as the *non-collusion assumption*. We use Goldberg's IT-PIR scheme [7] as it is robust against failing and Byzantine nodes, both of which exist in P2P networks.

DHTs. A file that is to be stored in a structured P2P network has a collision-resistant one-way hash, typically of its contents, as its identifier. A node on structured P2P networks also has a truncated one-way hash as its identifier; such a node stores all files whose identifiers, when truncated, are equal to the identifier of the node. Each node in a structured P2P network maintains a *routing table*, which consists of node identifiers and their routing information for a small number of other nodes in the DHT. In order to search for or insert a file, nodes obtain the routing information for the relevant node identifiers by querying one of its neighbours and then querying one of that neighbour's neighbours, and so on, through iterative routing. DHTs guarantee that for a network of size n, any node can reach any other node in at most $O(\log n)$ steps.

Censorship-resistant publishing. Censorship-resistant publishing (CRP) systems, such as Waldman and Mazières' Tangler [16], Waldman et al.'s Publius [17], and Vasserman et al.'s CROPS [15], support publishing and retrieving documents in the face of a censor that may attempt to take down prohibited content. CRP systems are built atop DHTs formed by machines that are located across different administrative regions. However, DHTs by themselves do not ensure the confidentiality, integrity, or availability of the content stored on them; CRP systems include mechanisms to publish and retrieve documents while providing confidentiality, integrity, and availability. We focus on CROPS as a use case for our system.

CROPS first encrypts a document to preserve its confidentiality. Additionally, encryption supports node operators in *plausibly denying* knowledge of the content they store. CROPS helps users easily discover content on the system through keywords. As the client's file retrieval query includes keyword hashes, the censor may legally oblige the node operator to reveal all queries that it received. The censor may then determine if any of the keyword hashes in a query match those of prohibited keywords [8, 9]. It may also require the node operator to report the network addresses of all users, and possibly harm the users who attempted to fetch prohibited content. Thus, simply providing plausible deniability over *stored* content is insufficient; node operators should also be prevented from learning what content is being *retrieved*.

3 ROBUST DHTS

DHTs have been known to be vulnerable to many attacks that prevent a user from obtaining a copy of a file. Existing CRP systems do not consider these attacks, and thus, do not integrate defences that have been proposed against such attacks. A censor can easily exploit these attacks, which we motivate below, to break the availability of files stored on the overlay CRP system. For instance, a censor can direct its new nodes to join at addresses that are close to a target honest node, and restricts its view of the network, by redirecting all routing requests to the these new nodes. Simply randomizing new nodes' identifiers is insufficient, as a censor can repeatedly rejoin an incoming node until it is close to a target honest node.

Forming quorums. Several researchers [3, 6] have proposed protocols to allocate joining nodes into quorums, such that by repeatedly rejoining, any malicious nodes do not increase their chances of being cast into a group with a majority of other such malicious nodes. Awerbuch and Scheideler [1] propose a cuckoo rule joining strategy, wherein incoming nodes cause existing nodes at nearby addresses to be kicked out (or cuckooed) to other addresses. They show that for a given *global* bound on the ratio of malicious to honest nodes (say ϵ), their strategy results in approximately equally sized quorums of $s = O(\log n)$ nodes, where the ratio of malicious to honest nodes in each quorum is upper bounded by a value greater than ϵ. Sen and Freedman [12] propose a *commensal cuckoo joining strategy*, which modifies the cuckoo rule joining strategy so that the network can withstand larger fractions of adversary-controlled nodes. Their strategy also supports setting the desired average quorum size beforehand, independent of the network size.

Robust routing across quorums. To resolve routing requests across quorums, nodes in a quorum can reply back with the network addresses of the nearest quorum to the target address, akin to iterative routing in regular DHTs. However, malicious nodes in such quorums may respond back with incorrect routing table entries, or inundate honest nodes in other quorums with fake requests. Young et al. [18] propose two efficient robust communication protocols (RCP) that either detect or prevent spamming attacks from nodes in other quorums by using a threshold signature scheme. In order to communicate with a node in a target quorum, the client node must obtain a time-stamped proof of robust communication from one of the neighbours of that quorum, in the form of a threshold signature of the neighbouring quorum.

Query privacy (QP) across quorums. The aforementioned protocols only provide *integrity* guarantees over the routing information within a quorum-based DHT. They do not provide *confidentiality* of the routing query content; the node address being queried is known to nodes on the path to the target node and to the target node itself. The censor may thus compel all nodes in its region, as they may serve as *in-path* nodes to other nodes, to reveal the node addresses looked up by users in its region. Backes et al. [2] improve on Young et al.'s protocols by hiding the queried key from all in-path nodes, thereby providing query privacy (QP).

Their protocols use oblivious transfer (OT) [11] in which the queried node sends the entry-wise encrypted data store over which the query is to be conducted back to the client. To privately retrieve a file stored in the file store of the target quorum, Backes et al. suggest optionally extending their protocols with one more hop to the target quorum. However, doing so would require sending the entire encrypted file store back ($\mathcal{D}$ bytes) to the client, resulting in a large communication complexity. Whereas, for a quorum of s nodes, our scheme only has a complexity of $2s\sqrt{\mathcal{D}}$. In Table 1 we contrast the properties provided by Young et al., Backes et al., and our schemes to DHT clients and compare the communication complexity for Backes et al.'s and our scheme in providing private file retrieval.

4 THREAT MODEL

We model the censor as controlling a small fraction of Byzantine nodes in the P2P network. For a network with n honest nodes, the

TABLE 1: **Comparison of private and robust DHT communication with previous work. $\mathcal{D}$ is the total size (in bytes) of the files stored at each node in an s-node quorum. RR = robust routing; QP = In-path query privacy; PR = availability and, if so, communication complexity, of private file retrieval.**

System	RR	QP	PR
Base DHT	-	-	-
Young et al.'s RCP [18]	●	-	-
Backes et al.'s QP [2] w/o last hop	●	●	-
Backes et al.'s QP [2] w/ last hop	●	●	$\mathcal{D}$
DHTPIR (our system)	●	●	$2s\sqrt{\mathcal{D}}$

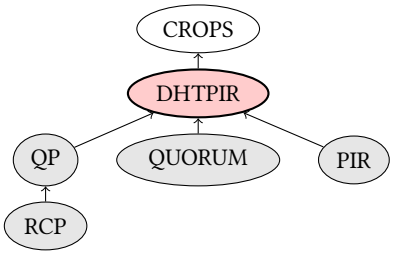

FIGURE 1: **The API interfaces required and provided by our DHTPIR interface. Red labels our contribution. Grey labels the layers of API required by our interface. White labels an existing system which our interface can support.**

censor may insert up to $\epsilon \cdot n$ malicious nodes, where $\epsilon \approx 0.02$; it can remove these nodes and have them rejoin the network, which would allow the node to join a different quorum. The censor's Byzantine nodes may respond incorrectly to routing or content retrieval requests. All Byzantine nodes within a quorum may collude and share the latter requests, in an attempt to reconstruct a client node's query. These nodes may also simply fail to reply to these requests or inundate honest nodes with them. The censor may monitor network traffic in its region of influence. It can also search for sensitive content by posing as a client and can map files to virtual addresses of nodes that store them.

The censor's goals are to determine which document was retrieved in a particular content retrieval request and to prevent the operator from responding correctly to a request. Our interface provides the following guarantees in the face of this adversary:

- Minimizing data within the content of document retrieval queries — *honest* server operator nodes and small coalitions of Byzantine nodes within a quorum do not learn which document was retrieved.
- Correct responses to document retrieval requests — a document retrieval request will be answered correctly, even if censor-controlled nodes provide incorrect or no responses.

5 OUR DESIGN

Our interface can be used within censorship-resistant publishing systems, such as Vasserman et al.'s CROPS [15], to insert files into quorums and to search quorums privately for files. Algorithms for both of these operations in the DHTPIR interface invoke existing protocols in layers, depicted in Figure 1.

Both of these algorithms expect at least an identifier for the file to be inserted or searched. They use this identifier to locate the file within the DHT (as in regular DHTs), with the exception that in our case, we wish to locate all nodes in a quorum; that is, a *set* of nodes in the DHT, all of which will store the file. Specifically, both algorithms use Backes et al.'s QP algorithms to obtain network addresses of the target quorum as well as cryptographic information that serves three purposes: for the quorum to verify the client's communication request (a cryptographic proof of communication), for the client to communicate confidentially with nodes in the quorum (node encryption keys) as well as verify threshold signatures over responses (quorum verification key).

For file retrieval operations, each quorum uses a perfect hash function (PHF) to index files in its database. A client node first retrieves the PHF from one of the quorum's nodes and then computes the index of the desired file identifier. It computes and encrypts one PIR query vector for each node of the quorum, following the aforementioned Goldberg's PIR protocol [7]. Each server node returns a PIR response, which is used by the client to reconstruct the file.

Intuitively, for inserting a file into the target quorum, the client needs to send its copy to each node in the target quorum. To preserve the confidentiality and integrity of the file content, consider a baseline wherein the client encrypts the file to each of the target nodes. As an optimization, we send the encrypted file to one of the target nodes, and delegate it to forward the file to all other nodes.

We account for Byzantine server and client nodes in our file retrieval and insertion algorithms. For instance, a Byzantine server node may return an invalid PHF or simply drop a file that it is delegated to forward to other nodes. Therefore, the delegate node returns a threshold signature over a hash of the inserted file or PHF to the client. The client verifies this signature, using the quorum verification key it obtained from the neighbouring quorum. This quorum also shares a cryptographic proof of communication that is used to prevent fake file insertion and retrieval requests. Additionally, with a given proof, (Byzantine) clients are rate limited to only send one file insertion and one file retrieval request.

We parameterize the threshold for Goldberg's IT-PIR scheme such that Byzantine nodes in a quorum cannot reconstruct a PIR response through collusion. Similarly, we parameterize the threshold for Young et al.'s threshold signature scheme, such that Byzantine nodes cannot produce enough valid signature shares through collusion. Furthermore, these Byzantine nodes cannot prevent the reconstruction of a threshold signature by simply failing to respond. Finally, we ensure that honest nodes can efficiently reconstruct the desired file, when a given fraction of nodes per quorum is Byzantine.

In our security analysis, which can be found in the extended version of our paper [10], we find that the fractions of Byzantine nodes in each quorum should be strictly less than 0.25 to satisfy the requirements of Young et al.'s robust quorums, and Goldberg's IT-PIR scheme. We extend Sen and Freedman's [12] simulation and we estimate the maximum value of ϵ that can allow only $b_0 < 0.25$ of each quorum to be malicious. When larger networks ($n \geq 2^{18}$) were configured with an average quorum of 20 nodes, the maximum observed quorum size was less than 75, and the network can withstand ϵ up to 0.02.

6 EVALUATION

In DHTPIR, chunks of documents are stored as fixed-size L-byte files. The communication and computation costs for retrieving a file for our system must scale when the number of nodes (*i.e.*, users) that connect to the P2P network (n) increases, as well as when the number of chunks that are stored in the network ($\mathcal{F}_N$) increases. We analysed our protocol in comparison with that of Backes et al. [2], and found that we can expect our protocol to require a smaller communication complexity for quorums than Backes et al. when $\mathcal{F}_N > \frac{sn}{L}$, where s is the number of nodes in a quorum. This holds for most reasonable values of $\mathcal{F}_N$, as typically $s \ll L$. For further details, see the extended version of our paper [10]. To evaluate this complexity empirically, we have implemented a simulation of our system. Our simulation code and output, as well as additional graphs, can be found at https://git-crysp.uwaterloo.ca/dhtpir/simulations.

In our simulation, one node serves as the client and creates $\mathcal{F}_N$ randomly generated document chunks of size 1 KiB. It stores each chunk on the quorum identified by the chunk's ID, then retrieves each chunk from the network. Different subprotocols within the system (namely, DHT, RCP, QP, OT, and PIR) are simulated as individual layers. We examine the case where systems have $q = 100$ quorums, each containing $s = 10$ nodes. The RCP layer implements Young et al.'s [18] RCP-II protocol. The QP layer implements Backes et al.'s QP-II protocol. We examine both the case where Backes et al.'s protocol includes OT at the last hop quorum (which we term "RCP+QP+LastHop", or simply "LastHop"), and the case where it does not (which we term "RCP+QP"). Finally, we use the following estimates of network and computation speeds. We set the network bandwidth to 50 Mb/s, and the round-trip time (RTT) to 150 ms. We estimate the PIR computations to run at 4 GB/s, whereas encryption operations run at 1 GB/s (about 3 cycles/byte for AES-NI).

We highlight the comparison of throughputs between DHTPIR and LastHop, which do protect the contents of queries, as well as RCP+QP, which does not, in Figure 2. Specifically, Figure 2 measures the number of simultaneous clients' file retrieval requests that can be handled by the entire system at once. This encompasses both request routing and file retrieval. This measurement is tracked as the number of documents per quorum ($\mathcal{F}_Q = \mathcal{F}_N/q$) increases. Note that in LastHop, as only a single delegate node in the target quorum is needed to compute the OT response for any given request (and computing threshold signatures is much faster than the OT computation), multiple nodes within a quorum can process distinct OT requests in parallel. For DHTPIR, each node in a quorum participates in computing the PIR response for a single PIR request; however, adding additional cores to nodes does allow for parallel processing of multiple distinct PIR requests at once.

We can see in Figure 2 that DHTPIR preforms the same regardless of the number of cores until $\mathcal{F}_Q \approx 2000$. In fact, it stays network-limited until this point, and after this point, adding another core to nodes in the system results in a significant improvement in throughput. This can be extrapolated to expand with database size, and thus, DHTPIR affords opportunities to optimize its throughput for nodes that store large databases through multi-core processing. Even though individual nodes within a quorum in LastHop can process different OT requests in parallel, LastHop incurs a large

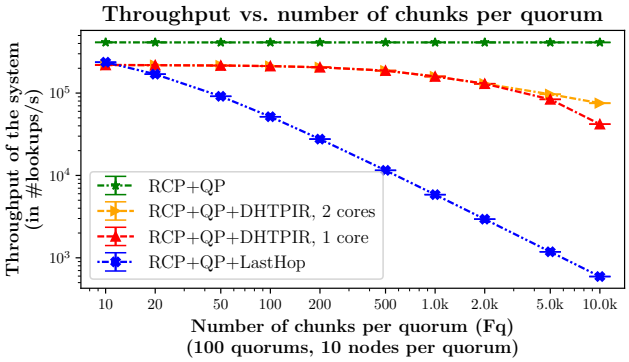

FIGURE 2: **Measurement of the throughput of DHTPIR and Backes et al.'s system, as the number of chunks stored in each quorum increases, averaged over 25 simulation runs. Note the logarithmic axes.**

reduction in throughput as compared to RCP+QP, proportional to the size of the database ($F_Q \cdot L$). LastHop's low throughputs here are largely the result of the bandwidth required to transmit large amounts of data necessary for that protocol to implement file retrieval, and additional cores would not be able to increase these throughputs. DHTPIR achieves a throughput that is about two orders of magnitude larger than LastHop, and this gap widens with database size. Given this, our simulation provides us reasonable grounds to believe in the efficiency of a DHTPIR implementation.

7 CONCLUSION

Censorship-resistant publishing systems are built atop DHTs and enable users to store sensitive documents onto multiple server nodes in different administrative regions, such that a censor cannot easily take down or tamper with the published content. Server node administrators may be compelled by censors to reveal the documents retrieved by a given user. We propose an interface that uses information-theoretic private information retrieval (IT-PIR) to prevent node operators from being exposed to information about which document was retrieved.

We integrate existing work on hardening peer-to-peer networks so that quorums of nodes only contain at most a certain fraction of Byzantine nodes. Our key insight lies in using quorums as a coalition of server nodes that store a set of files over which IT-PIR queries can be performed. We simulate our system and find that its throughput is two orders of magnitude higher than the closest related work and scales reasonably with the database size. We hope that our design spurs further research and development efforts into building robust censorship-resistant publishing systems.

ACKNOWLEDGMENTS

We thank the Royal Bank of Canada and NSERC grant CRDPJ-534381 for funding this work. This research was undertaken, in part, thanks to funding from the Canada Research Chairs program. This work benefitted from the use of the CrySP RIPPLE Facility at the University of Waterloo.

REFERENCES

[1] Baruch Awerbuch and Christian Scheideler. 2009. Towards a Scalable and Robust DHT. *Theory of Computing Systems* 45, 2 (2009), 234–260. https://doi.org/10.1007/s00224-008-9099-9

[2] Michael Backes, Ian Goldberg, Aniket Kate, and Tomas Toft. 2012. Adding Query Privacy to Robust DHTs. In *7th ACM Symposium on Information, Computer and Communications Security* (Seoul, Korea) *(ASIACCS '12)*. 30–31.

[3] Ingmar Baumgart and Sebastian Mies. 2007. S/Kademlia: A practicable approach towards secure key-based routing. In *2007 International Conference on Parallel and Distributed Systems*. IEEE, 1–8.

[4] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. 1998. Private Information Retrieval. *Journal of the ACM* 45, 6 (1998), 965–981. https://doi.org/10.1145/293347.293350

[5] Roger Dingledine, Nick Mathewson, and Paul Syverson. 2004. Tor: The Second-generation Onion Router. In *Proceedings of the 13th USENIX Security Symposium* (San Diego, CA). 21–21.

[6] Amos Fiat, Jared Saia, and Maxwell Young. 2005. Making Chord Robust to Byzantine Attacks. In *Proceedings of the 13th Annual European Conference on Algorithms* (Palma de Mallorca, Spain) *(ESA'05)*. Springer-Verlag, Berlin, Heidelberg, 803–814.

[7] Ian Goldberg. 2007. Improving the Robustness of Private Information Retrieval. In *2007 IEEE Symposium on Security and Privacy (SP '07)*. IEEE, 131–148.

[8] Anne Henochowicz. 2015. The Human Side of Censorship: Keyword Filtering and Censorship Directives on the Chinese Internet. In *Proceedings of the 5th USENIX Workshop on Free and Open Communications on the Internet (FOCI 2015)*. Washington D.C., USA. https://www.usenix.org/sites/default/files/conference/protected-files/foci15_slides_henochowicz.pdf

[9] Jeffrey Knockel, Masashi Crete-Nishihata, Jason Q. Ng, Adam Senft, and Jedidiah R. Crandall. 2015. Every Rose Has Its Thorn: Censorship and Surveillance on Social Video Platforms in China. In *Proceedings of the 5th USENIX Workshop on Free and Open Communications on the Internet (FOCI 2015)*. Washington D.C., USA. https://www.usenix.org/system/files/conference/foci15/foci15-paper-knockel.pdf

[10] Miti Mazmudar, Stan Gurtler, and Ian Goldberg. 2021. Do you feel a chill? Using PIR against chilling effects for censorship-resistant publishing. Cryptology ePrint Archive, Report 2021/1195. https://eprint.iacr.org/2021/1195.

[11] Moni Naor and Benny Pinkas. 2001. Efficient Oblivious Transfer Protocols. In *Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms* (Washington, D.C., USA) *(SODA '01)*. Society for Industrial and Applied Mathematics, USA, 448–457.

[12] Siddhartha Sen and Michael J. Freedman. 2012. Commensal Cuckoo: Secure Group Partitioning for Large-Scale Services. *ACM SIGOPS Operating Systems Review* 46, 1 (2012), 33–39. https://doi.org/10.1145/2146382.2146389

[13] Adam Stubblefield and Dan S. Wallach. 2002. *Dagster: Censorship-Resistant Publishing Without Replication.* Technical Report. Rice University.

[14] Eugene Y. Vasserman, Victor Heorhiadi, Nicholas Hopper, and Yongdae Kim. 2012. One-Way Indexing for Plausible Deniability in Censorship Resistant Storage. In *Proceedings of the 2nd USENIX Workshop on Free and Open Communications on the Internet*. USENIX, Bellevue, WA.

[15] Eugene Y. Vasserman, Victor Heorhiadi, Yongdae Kim, and Nicholas J. Hopper. 2011. *Censorship resistant overlay publishing*. Technical Report 11-027. University of Minnesota.

[16] Marc Waldman and David Mazières. 2001. Tangler: A Censorship-Resistant Publishing System Based On Document Entanglements. In *Proceedings of the ACM Conference on Computer and Communications Security*. ACM, 126–135.

[17] Marc Waldman, Aviel D. Rubin, and Lorrie Faith Cranor. 2000. Publius: A robust, tamper-evident, censorship-resistant, web publishing system. In *Proceedings of the 9th USENIX Security Symposium*. USENIX, 59–72.

[18] Maxwell Young, Aniket Kate, Ian Goldberg, and Martin Karsten. 2013. Towards Practical Communication in Byzantine-resistant DHTs. *IEEE/ACM Transactions on Networking* 21, 1 (2013), 190–203.