Harpocrates: Anonymous Data Publication in Named Data Networking

Md Washik Al Azad
University of Nebraska at Omaha
malazad@unomaha.edu

Reza Tourani
Saint Louis University
reza.tourani@slu.edu

Abderrahmen Mtibaa
University of Missouri–Saint Louis
amtibaa@umsl.edu

Spyridon Mastorakis
University of Nebraska at Omaha
smastorakis@unomaha.edu

Abstract

Named-Data Networking (NDN), a prominent realization of the Information-Centric Networking (ICN) vision, offers a request-response communication model where data is identified based on application-defined names at the network layer. This amplifies the ability of censoring authorities to restrict user access to certain data/websites/applications and monitor user requests. The majority of existing NDN-based frameworks have focused on enabling users in a censoring network to access data available outside of this network, without considering how data producers in a censoring network can make their data available to users outside of this network. This problem becomes especially challenging, since the NDN communication paths are symmetric, while producers are mandated to sign the data they generate and identify their certificates. In this paper, we propose *Harpocrates*, an NDN-based framework for anonymous data publication under censorship conditions. *Harpocrates* enables producers in censoring networks to produce and make their data available to users outside of these networks while remaining anonymous to censoring authorities. Our evaluation demonstrates that *Harpocrates* achieves anonymous data publication under different settings, being able to identify and adapt to censoring actions.

Keywords

Producer Anonymity, Censorship, Named-Data Networking, Information-Centric Networking

1 Introduction

Preserving the anonymity of users that generate and share data (*e.g.,* pictures, videos, messages) with others is crucial especially in scenarios where the safety and freedom of users may be in danger (*e.g.,* authoritarian regimes) [36]. In such scenarios, authorities, such as governments, Internet Service Providers (ISPs), and other organizations, may restrict access to certain websites and block the operation of applications that allow users to publish their data on the Internet [24]. The ultimate goal of these authorities is to limit access to data that they do not consider favorable and avoid having non-favorable data (*e.g.,* videos of protests, pictures of illegal practices) be published by their users on the Internet. For example, during protests in a certain country, protesters may take pictures or videos that show law enforcement personnel attempting to violently suppress these protests. The government may restrict protesters from uploading this data to popular hosting (*e.g.,* YouTube and Vimeo), news (*e.g.,* CNN and BBC), and social media (*e.g.,* Facebook and Instagram) websites. Even in cases that users find ways to upload their data on the Internet (*e.g.,* on websites not blocked by the government), the government in cooperation with local ISPs may be able to identify the citizen(s) that uploaded the data and imprison them. At the same time, hosting, news, and social media websites have vested interest in verifying the authenticity of the uploaded data before making it available to their users without compromising the anonymity of the data producer.

This scenario highlights the following fundamental questions when it comes to publishing data under censorship: *(i) how can citizens/users that produce data within oppressive countries and organizations (censoring networks) publish this data on the public Internet (non-censoring networks)? and (ii) how can the produced data be published and authenticated on the public Internet while its producers remain anonymous to the oppressive countries and organizations, which could threaten the producers' safety and well-being?* Solutions to tackle these issues have been proposed in the context of the IP-based network architecture [11, 22, 40, 44, 45, 48].

Over the last decade, the direction of Information-Centric Networking (ICN) [3] and its prominent realization, Named-Data Networking (NDN) [47], have attracted attention by the research community. NDN features a request-response communication model, where requests that identify the data by application-defined names are forwarded towards data producers. NDN possesses the privacy friendly features of not containing specific source and destination addresses in its packets. However, the use of semantically rich names at the network layer amplifies the ability of censoring authorities

This paper has been accepted by the ACM Symposium on Access Control Models and Technologies (SACMAT) 2022. The definite version of this work will be published by ACM as part of the SACMAT conference proceedings.

to restrict access to non-favorable data and monitor what data their users request.

Several solutions have been proposed to alleviate this issue [5, 6, 10, 29, 42], focusing on how users in a censoring network can access data available in an non-censoring network. However, these solutions did not consider how producers residing in a censoring network can make their data available to users outside of this network. This problem in NDN is especially challenging due to: (i) the retrieval process, initiated by requests that carry the names of the data to be retrieved, empowers censoring authorities to drop requests for data produced within the censoring network at the border of this network; (ii) the symmetry of the communication model (*i.e.,* each response follows the same network path back to a requester as the corresponding request) enables censoring authorities to analyze requests and block the corresponding data on the way back to the requester; and (iii) as a by-product of (i), (ii), and the NDN semantically meaningful naming, producers in censoring networks cannot advertise the data they produce, since this would enable censoring authorities to directly link the generated data to them and cannot be reachable from outside of censoring networks, since censoring authorities can easily drop incoming requests with non-favorable or unknown data names.

To tackle these challenges, we propose *Harpocrates*[1], an NDN-based framework for anonymous data publication, which enables producers in censoring networks to produce, upload, and make their data available to users outside of these networks while remaining anonymous to censoring authorities. *Harpocrates* makes the following contributions:

- It takes advantage of communication channels and applications that operate legally within the censoring network as well as a decoy routing approach [22] to publish data in a peer-to-peer fashion, maximizing the collateral damage for censoring authorities;
- It features mechanisms for producers to identify censoring activities and adapt their data publication process to such activities. This ensures that the data will be successfully uploaded to a network of trusted proxies in order to become available to users outside of the censoring network;
- It features a secure delegation mechanism between producers and proxies, preventing censoring authorities from being able to link the generated data back to producers. As a result, proxies can make the data available outside of the censoring network on behalf of producers while preserving the producers' anonymity and, at the same time, enabling users to verify data authenticity.

To the best of our knowledge, *Harpocrates* is among the first attempts in NDN/ICN environments to tackle the problem of making data generated within a censoring network available outside of this network without compromising the producer's anonymity.

[1] *Harpocrates* was the god of secrets and confidentiality according to the ancient Greek mythology.

2 Background and Prior Work

In this section, we give a brief background of the NDN architecture and discuss related work in both IP and NDN/ICN environments.

2.1 Named-Data Networking

NDN [47] features a receiver-driven model that leverages application-defined semantically meaningful naming for communication purposes. In NDN, *consumer applications* send requests for data, called Interest packets. Interests are forwarded based on their names towards *data producer applications*, which send Data packets that contain the requested data back to consumers.

For the realization of the NDN communication model, NDN routers maintain three data structures: (i) a Forwarding Information Base (FIB), which consists of name prefixes along with a number of outgoing interfaces for each prefix and is used for Interest forwarding; (ii) a Pending Interest Table (PIT), which stores Interests that have been recently forwarded but have not retrieved data yet; and (iii) a Content Store (CS), where retrieved Data packets are cached to satisfy future requests for the same data.

NDN is based on three fundamental principles: (i) *identifying network-layer packets through application-defined, semantically meaningful names*–NDN carries network-layer packets that contain application-defined names; (ii) *securing data directly at the network layer*–each network-layer Data packet carries the signature of its producer, which cryptographically binds the actual data to the packet's name and secures the data at rest and in transit across the network, along with signature related information that specifies the producer's certificate or public key [39]; and (iii) *a stateful forwarding plane:* forwarded Interests leave state at each router, while Data packets follow the reverse path of the corresponding Interests, consuming the state at each router.

2.2 Prior Work on Censorship Circumvention and Anonymity

2.2.1 *IP-based Censorship Circumvention and Anonymity*

Tor [40] is the most popular anonymity network, which uses an overlay of relays to provide identity anonymity and unlinkability. Extensive research has been conducted on various facets of Tor [11, 44, 45]. However, using layers of encryption and decryption to secure the data imposes considerable overhead and impacts communication latency. Tor's high latency and its vulnerability to active probing [13] motivated the design of an alternative approach, decoy routing [22]. Decoy routing is an in-network censorship circumvention platform, where a set of decoy routers participate in relaying the traffic outside of a censoring network. Several flavors of decoy routing have been proposed to enhance the seminal design through decoy placement optimizations [35] routing optimizations based on game theory [31], mimicking access patterns to non-censored websites [7], and routing

asymmetries [30]. Another censorship evading direction includes mimicking the traffic profiles of non-censored, innocuous applications [15, 46]. The community has also investigated frameworks that utilize public Content Delivery Networks (CDNs) to access censored data under the assumption that blocking data hosted on these CDNs will cause collateral damage, since innocuous data publishers will be disrupted [48].

2.2.2 NDN/ICN-based Censorship Circumvention and Anonymity The state-of-the-art in NDN/ICN censorship circumvention and anonymous communication is categorized into proxy-independent and proxy-based techniques [42]. In this realm, the use of steganography, where data and a cover file need to be combined before publication, was among the first proposals [5]. Users obtain the necessary data decoding information through a secure back channel. This scheme imposes considerable communication overhead, which impacts its scalability. Techniques, such as homomorphic encryption, have been also proposed in a publish-subscribe design to provide privacy for user requests [16].

Tor has inspired proxy-based solutions [10, 23], where layers of encryption between users and a network of proxies are used for anonymity. CoNaP [25] takes a similar approach, where a user encrypts and signs the names of Interests for authenticity. However, this signature reveals the user identity and compromises its anonymity [32]. To reduce the cost of a symmetric key cryptosystem, which needs to be carried on a per-packet basis, lightweight coding techniques, including random linear network coding [38] and Huffman coding [41], have been proposed. PrivICN [6] is another proxy-based scheme that enables cache utilization. By employing proxy re-encryption, PrivICN enables cached data in the censoring network to be used by multiple users. However, cache hits in the censoring network introduce information leakage and undermine user anonymity. A decoy routing approach was also proposed for traffic redirection [29], where a user informs a decoy router through a covert channel to redirect its requests to the covert rather than the decoy destination. Finally, an Attribute-Based Signature scheme for NDN was proposed [32]. However, this scheme focuses on anonymizing a producer's signatures, without considering any other aspects of the anonymous data publication process.

How does *Harpocrates* differ from prior work? While the majority of existing NDN/ICN approaches have focused on enabling consumers within a censoring network to reach producers in non-censored networks to download data, very few designs have considered the problem of anonymous data publication in NDN/ICN. Such designs primarily focus on signature anonymization [32] or rely on onion routing, which requires multiple, costly layers of encryption [23]. Our work enables producers in a censoring network to publish (upload) their data to consumers outside of this network in an anonymous manner without the need for multiple layers of encryption.

Table 1: Summary of notations.

Notation	Description
P, Q	Big prime numbers such that $P = 2Q + 1$
$\mathbb{Z}_Q^*, \mathbb{Z}_P^*$	Multiplicative groups of integers of order Q and P respectively
$\mathbb{G}_Q, \mathbb{G}_P$	Cyclic groups of order Q and P respectively
$\mathbb{G}_S$	Schnorr group (large prime-order subgroup of $\mathbb{Z}_P^*$)
g	Generator of a sub-group of $\mathbb{G}_P$ of order Q
$ZQrand()$	Random number generator in $\mathbb{Z}_Q^*$
(PK_X, PR_X)	X's public and private signing key pair
S_{XY}	Symmetric key shared between X and Y
$H() : \{0,1\}^* \rightarrow \mathbb{Z}_Q^*$	Cryptographic hash function with digest $\in \mathbb{Z}_Q^*$
W	Warrant for proxy signature delegation
M	Message to be signed
$\|\|$	Concatenation operator
$\equiv$	Congruence operator

3 Model and Assumptions

In this section, we present our system and network model, our design assumptions, our threat model, and the goals of the *Harpocrates* design. Table 1 includes the notations we use in the rest of this paper.

3.1 System and Network Model

We consider a censoring network and a set of proxies that make data available to consumers outside of this network (Fig. 1). Our system model consists of the following actors:

- **Producer:** An entity in the censoring network that wishes to anonymously publish data (potentially consisting of several network-layer Data packets) outside of this network.
- **Consumer:** An entity outside of the censoring network interested in the data generated by the producer.
- **Peers:** Entities (in the censoring network) subscribed to a peer-to-peer application that operates "legally" in the censoring network. The producer is a peer running this application.
- **Collaborating peers:** Peers selected by the producer to help make the data generated by the producer available outside of the censoring network.
- **Censoring nodes:** Entities deployed by ISPs, governments, or other stakeholders in the censoring network to detect and block attempts to publish data outside of this network.
- **Selected proxy:** A trusted entity outside of the censoring network that collects, reconciles, and publishes the data on behalf of the anonymous producer, so that consumers outside of the censoring network can access this data.
- **Collaborating proxies:** Trusted entities outside of the censoring network that receive censored data from the collaborating peers and send this data to the selected proxy.

We illustrate our system through a running example in Fig. 1. The producer selects a set of collaborating peers (subset of the overall peers) and shares with them pieces of the generated data. However, these pieces can be intercepted and blocked by censoring nodes on their way to the collaborating peers. A collaborating peer receiving a data piece will send it towards a collaborating proxy. Each collaborating proxy will eventually forward the received data to the selected proxy.

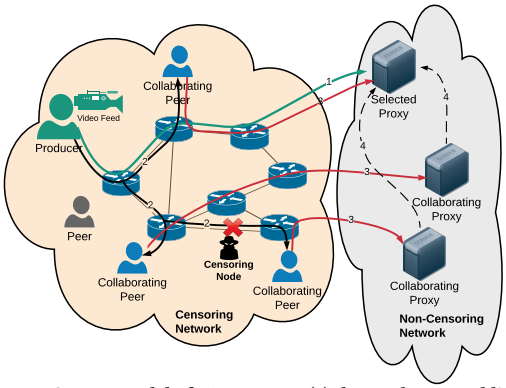


Figure 1: System model of *Harpocrates*: (1) the producer establishes a covert channel with a selected proxy; (2) the producer shares pieces of data with collaborating peers in a peer-to-peer fashion, while censoring nodes may intercept these pieces; (3) collaborating peers push the data towards collaborating proxies outside of the censoring network in manners that prevent traffic analysis attacks; and (4) collaborating proxies share the data with the selected proxy that makes it available to consumers outside of the censoring network.

3.2 Assumptions

We assume that producers in the censoring network do not advertise their data to protect their anonymity. We also assume that producers are not reachable from outside of the censoring network, thus they cannot directly upload their data to consumers outside of this network. This is a fair assumption considering the symmetric, name-based nature of NDN communication. This makes it trivial for censors to block requests or responses for data produced in their network and for entities within their network to ensure that censored data does not become available to the outside world.

We consider rational attackers with bounded capabilities, that is attackers who do not orchestrate large-scale brute force attacks or block all the communication in the censoring network. This is a fair assumption since pervasive blocking causes collateral damage [48]. We assume that neither collaborating proxies nor collaborating peers (selected by the producer) are malicious. This is a fair assumption since the majority of censorship circumvention tools leverage trust and reputation-based mechanisms to select entities playing key roles. For instance, in Tor, only trustworthy relay nodes can be selected as *entry guards* due to their importance in protecting user anonymity [12]. We assume the existence of an anonymous public-key certificate approach [20], which preserves the privacy of the producers' information in their certificates. We discuss directions to further augment producer anonymity in Section 8. Finally, we assume that symmetric and asymmetric cryptographic operations are secure.

3.3 Threat Model

In NDN, the use of names at the network layer can simplify data filtering, censorship, and violate the consumer and producer privacy. In this paper, we consider that a censoring authority can deploy active attackers and passive eavesdroppers across the censoring network to interrupt ongoing data publications from this network to the outside world or compromise producer anonymity. An active attacker can capture and modify transmitted packets, while a passive eavesdropper can analyze the captured packets. Deployed attackers may masquerade as different entities such as peers.

The primary objective of the censoring authority is to prevent producers in the censoring network from publishing data. Thus, the censoring authority may: *(i)* block the ISP's ingress Interests destined to producers; *(ii)* act as a man-in-the-middle to collect the requested data from the producers, compare it against a blacklist, and either drop the packets or relay them to the requester; *(iii)* deploy censoring nodes to interrupt the ongoing communication across peers by dropping the Interest and/or Data packets; and *(iv)* masquerade as a peer to interrupt the communication and compromise the producer's anonymity. We note that objectives *(iii)* and *(iv)* are different in the sense that in the former one, the attacker is an ISP node in the censoring network (*e.g.,* a router), while in the latter one, the attacker is one of the peers. While the focus of this work is enabling anonymous data publication rather than coping with traffic analysis attacks, we will briefly discuss potential traffic analysis countermeasures in Section 8 to thwart this category of attacks.

3.4 *Harpocrates* Design Goals

Harpocrates offers data producers–whether in a censoring network or not–to successfully publish their data (evade censorship) while preserving their privacy and data integrity. *Harpocrates* has the following goals:

- **Anonymity and plausible deniability:** *Harpocrates* should preserve the producer's anonymity in the presence of different attackers. The attackers may interrupt the data publication, but should neither be able to reveal the producer's identity nor link the published data to the producer.
- **Integrity guarantees:** *Harpocrates* should guarantee the integrity of the published data without revealing the producer's identity. This is important as the producer delegates the publication of its data to a third party (selected proxy).
- **Reasonable overhead:** *Harpocrates* should incur reasonable communication and computation overhead on the involved actors. The cost of *Harpocrates* for the collaborating peers should be viable, while the producer should be able to publish its data with reasonably low latency.

4 Design Overview

In this section, we present an overview of *Harpocrates* (Fig. 2). In *Harpocrates*, the producer will first reach and securely delegate the data publication privilege to the selected proxy that will help preserve the producer anonymity. After the secure delegation phase, the producer will start the data uploading phase through a peer-to-peer mechanism to: (i) prevent the censoring authority from detecting abnormal amounts of data from a single peer being sent outside of the censoring network; and (ii) ensure that the data production cannot be linked back to the producer.

These phases are facilitated through the use of decoy routing techniques [21], which we briefly discuss in the rest of

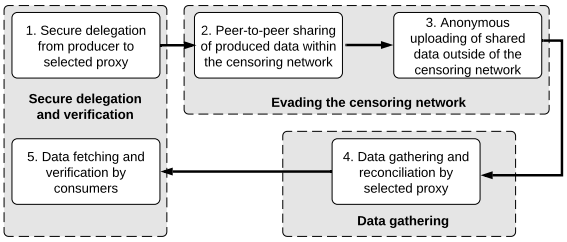


Figure 2: Overview of the *Harpocrates* design.

this section and provide details in Sections 5 and 6. Different than in IP, decoy routing in *Harpocrates* is realized through decoy name prefixes (*i.e.,* prefixes that allow Interests from within the censoring network to be forwarded outside of this network). Benign information encoded in names help proxies identify Interests with decoy prefixes. Combined with the fact that NDN routers have direct access to the names of Interests, decoy routing in *Harpocrates* can be easily deployed outside of the censoring network, without requiring routers to search for signalling information at higher layers of the protocol stack (*e.g.,* transport or application layer) [21]. This makes the deployment of decoy routers flexible and simple, while decoy router assignments can change over time (*e.g.,* coordinated through routing protocols in non-censoring networks and selected based on placement strategies that maximize collateral damage [31]) to overcome known attacks against decoy routing (*e.g.,* routing around decoys [35]).

4.1 Secure Delegation Overview

In this phase, the producer employs decoy routing to select and reach one of the collaborating proxies, who will act as the selected proxy, outside of the censoring network. Subsequently, the producer securely provides delegation metadata and instructions to the selected proxy for data publication. The selected proxy, on receiving the metadata, accepts the data publication by returning a signed *commitment* to prove its involvement in data publication and enable the initiation of the proxy signature process (Definition 4.1). We use a warrant-based proxy signature [2] based on the difficulty of the discrete logarithm problem.

Definition 4.1. [Proxy Signature] *Proxy signature is a cooperative digital signing scheme [27], in which an original signer (data producer in our case) delegates its right of digitally signing a message to a proxy. Such a delegation allows verifiers (consumers in our case) with knowledge of the signer's and proxy's public keys to validate signed messages. Proxy signature schemes are categorized into full delegation, partial delegation, and delegation by warrant. A warrant includes metadata, such as delegation scope information to authorize the proxy to sign on behalf of the original signer.* □

The producer then generates the required credentials for proxy signature and securely sends them to the selected proxy for data signing and publication. In *Harpocrates*, we use Schnorr group (Definition 4.2) and Schnorr signature [34].

Definition 4.2. [Schnorr Group] *Given two large primes Q and P, where $P = rQ+1$, $r \in \mathbb{Z}_Q^*$ and $\mathbb{Z}_Q^*$ is the multiplicative group of integers mod Q, choose $1 < h < P$, such that $h^r \not\equiv 1$ mod P, then $g = h^r$ generates a Schnorr group ($\mathbb{G}_S$). $\mathbb{G}_S$ is a subgroup of $\mathbb{Z}_P^*$, the multiplicative group of integers mod P of order Q [34].* □

4.2 Anonymous Data Uploading Overview

Overall, the *Harpocrates* communication design consists of two main steps: (i) *evading the censoring network* by sending all producer's Data packets to collaborating proxies outside of censoring network without compromising producer's anonymity; and (ii) *gathering of all Data packets* by the selected proxy, reconstructing the original producer's data, and making this data available to consumers on the Internet.

To maximize collateral damage for the censoring authority, *Harpocrates* features a peer-to-peer mechanism, where the producer makes its data available through decoy routing towards proxies outside of the censoring network. Subsequently, Internet users can fetch the uploaded data from the proxies. The peer-to-peer mechanism leverages existing and allowed channels of communication (*e.g.,* gaming or local social media applications) in the censoring network. These allowed applications and communication channels are used to "hide" data transfers towards the collaborating peers, spreading the data uploading traffic across these peers in the censoring network. We note that having producers use decoy routing to directly reach collaborating proxies would result in significant volumes of traffic initiated by producers and traffic anomalies that can be detected by censors.

In NDN, communication among multiple parties (*e.g.,* for a multiplayer gaming application) is realized through a distributed synchronization protocol [26]. This protocol creates a multicast name prefix for communication among all the peers in a group (*e.g.,* users that play an online game as a team), ensuring that a request sent from one peer in this group will be received by all other peers. Through this synchronization process and the established multicast channel, collaborating peers can share information. However, this process can be infiltrated by the censoring network through the deployment of censoring nodes as routers and/or peers in the synchronization group to intercept traffic[2]. To cope with that, *Harpocrates* offers a data encryption mechanism during the communication among collaborating peers, ensuring that only selected collaborating peers will be able to decrypt the exchanged information.

5 Secure Delegation Design

In this section, we present the secure delegation phase (Fig. 3), including the *proxy commitment*, *signature generation*,

[2]The routers deployed as censoring nodes not only can intercept but also drop the exchanged traffic. However, this would cause significant collateral damage given that peers utilize allowed communication channels and applications as we further discuss in Section 8.

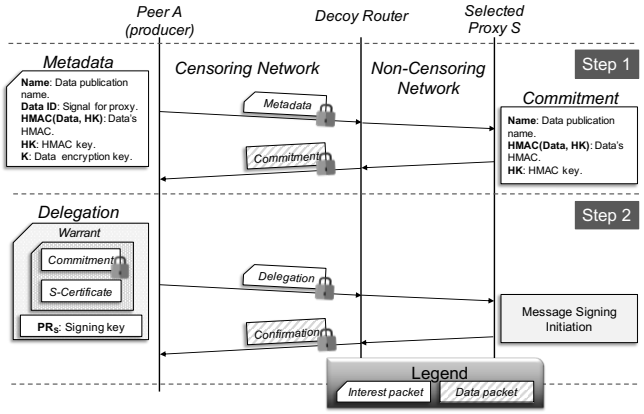


Figure 3: The secure delegation phase of *Harpocrates* has two steps. First, peer A (producer) provides delegation metadata to the selected proxy (S) for data publication and obtains the selected proxy's commitment. Subsequently, peer A provides the credentials for proxy signature to the selected proxy.

delegated message signing, and *signature verification* protocols. The goal is for the producer to get the selected proxy to commit publishing the data on behalf of the producer, while guaranteeing data integrity, confidentiality, and anonymity. As mentioned in Section 3.2, we assume that producers have anonymous public key certificates [20] to avoid revealing information about themselves through their certificates. In Section 8, we discuss approaches to augment the anonymity level that generic public key certificates provide.

5.1 Proxy Commitment

As shown in Step 1 of Fig. 3, the producer generates delegation metadata, including the *Data Name* under which the selected proxy should publish the data, the *Data ID* to signal the selected proxy of the related Data packets, the data *HMAC* (keyed-hash message authentication code) and its key (*HK*), and the data encryption key (*K*). The Data ID is a random string to be used as part of the data name, informing the selected proxy of the Data packets that belong to this particular data collection. The producer then securely (signed and encrypted) transmits the metadata to the selected proxy. We employ decoy routing to make the proxies reachable to the peers inside the censoring network. As stated in Section 3.3, the censoring authority blocks requests from outside of the censoring network to prevent leaking internal data. Thus, the producer and peers can only communicate with proxies by attaching information (*e.g.,* delegation metadata) to Interest packets and send them using decoy name prefixes.

Upon receiving the delegation metadata, the selected proxy ("proxy" in the rest of this section) uses the metadata to create the *commitment*, including the data name, its HMAC, and the HMAC's key. It then signs the commitment using its primary key pair, encrypts it using the shared session key, and returns it to the producer. The rationale behind enforcing the proxy to generate the commitment is to prevent a malicious proxy from altering the producer's data before publication. The use of warrant-based proxy signatures [2]

Protocol 1 Proxy Signature Generation by Producer A

Input: $\mathbb{G}_S, W, H()$, and $ZQrand()$.
Output: PR_S (Proxy S private key).
1: Choose signing private key $PR_A \in \mathbb{Z}^*_Q$.
2: Calculate corresponding public key $PK_A = g^{PR_A} \in \mathbb{Z}^*_P$.
3: Select $i = ZQrand()$.
4: Calculate $t = g^i \in \mathbb{Z}^*_P$.
5: Generate $W_h = H(W||t)$.
6: Calculate $PR_S = (W_h \times PR_A + i) \in \mathbb{Z}^*_Q$.
7: Store $< PR_S, W, t >$

requires the producer to generate a warrant and a signing key pair for the selected proxy–from its own asymmetric key. The generated signing key (delegated key pair) is different from the selected proxy's primary key pair and should be used for proxy signature.

5.2 Producer Signature Delegation

As shown in Step 2 of Fig. 3, peer A generates a warrant composed of the selected proxy's signed commitment, the proxy's certificate (corresponds to its primary key pair for commitment verification), and the producer's public key (Protocol 1). This information authorizes the selected proxy to sign on behalf of the producer and restricts the selected proxy from abusing the delegated authority (*e.g.,* altering the data or its name included in the commitment). Having the warrant generated, the producer needs to derive the proxy's delegated key pair through the proxy signature scheme.

Protocol 1 takes an agreed upon Schnorr group ($\mathbb{G}_S$), the hashing function ($H()$), and the warrant (W) as inputs and returns the private signing key (PR_S) of the selected proxy (proxy S); the delegated private key. Producer A initiates this process by choosing a Schnorr private signing key (PR_A) and generating the corresponding public verification key (PK_A) (Lines 1-2). To derive the selected proxy's signing key, A selects a random integer i in the multiplicative groups of integers of order Q and calculates t (Lines 3-4). It then generates the warrant's digest (W_h) using W and t (Line 5). In Line 6, A uses the warrant's digest (W_h), its private key (PR_A), and i to calculate the selected proxy's private signing key (PR_S). The equation in Line 6 shows the involvement of A's private key in generating the selected proxy's private signing key. Finally, A securely sends the generated private key (PR_S), the warrant (W), and t to S. The completion of Protocol 1 concludes the interactions between A and S.

5.3 Proxy Data Signing

Upon collecting all Data packets, proxy S executes Protocol 2 to sign the packets on behalf of producer A using the delegated private signing key (PR_S). Protocol 2 accepts the agreed upon Schnorr group ($\mathbb{G}_S$), the hash function ($H()$), a Data packet (message M), and the three-tuple S received from A ($< PR_S, W, t >$) and returns a signed packet.

Initially, S uses $\mathbb{G}_S$ and its private signing key (PR_S) to generate the corresponding public verification key (PK_S), which will be used by consumers to verify the delegated proxy signature on the Data packets. To ensure the validity

Protocol 2 Delegated Message Signing by Proxy S

Input: $\mathbb{G}_S, H(), M$, and $< PR_S, W, t >$.
Output: Signed Message.
1: Calculate proxy S public key $PK_S = g^{PR_S} \in \mathbb{Z}_P^*$.
2: Generate $W_h = H(W||t)$.
3: **if** $\left(PK_S \equiv (PK_A^{W_h} \times t) \in \mathbb{Z}_P^*\right)$ **then**
4: Select $r = ZQrand()$.
5: Calculate $k = g^r \in \mathbb{Z}_P^*$.
6: Generate $a = H(M||k) \in \mathbb{Z}_Q$.
7: Calculate $b = \left(r - (PR_S \times a)\right) \in \mathbb{Z}_Q$.
8: Store $< M, W, t, a, b >$.
9: **else**
10: Fail.
11: **end if**

of PK_S, S generates the warrant's digest (Line 2) and verifies its congruence with A's public key (PK_A) (Line 3). The correctness of the congruence in Line 3 shows the involvement of A's public key in generating the delegated public verification key (PK_S). We note that Lines 1-3 of Protocol 2 need to be executed once. Thus, the cost of executing these steps is negligible when amortized over multiple signing operations. To sign a Data packet (message M), S executes Lines 4-8 of Protocol 2–the signing process follows Schnorr signature. S selects a random integer $r \in \mathbb{Z}_Q^*$ and calculates its corresponding value k (Lines 4-5). It then uses k in generating the message digest a (Line 6). Using the private signing key (PR_S), the digest (a), and the random integer (r), S signs the message (Line 7) and stores the signature as a five-tuple ($< M, W, t, a, b >$) for the consumer's verification process.

5.4 Signature Verification

Protocols 1 and 2 enable consumers to validate the proxy's signatures and the delegation authorization, ensuring that S is certified by A. Protocol 3 details the verification process by accepting the Schnorr group ($\mathbb{G}_S$), the hash function ($H()$), and the five-tuple generated by S ($< M, W, t, a, b >$).

The consumer, verifying the selected proxy's signature, generates the warrant's digest (W_h) using warrant W and t from the signature (Line 1). The consumer then uses W_h and A's public key (PK_A) to derive the signature verification key (y). The amortized cost of extracting the signature verification key (y) will be negligible as Lines 1-2 will be executed once for a set of signature verification operations. After extracting y, the consumer executes Lines 3-4, which refer to the conventional Schorr signature verification process. Following Lines 5-9, the consumer accepts the signature if the received signature (a) matches the one that it generates (a_v) or rejects, otherwise.

Protocol 3 Signature Verification by Consumers

Input: $\mathbb{G}_S, H()$, and $< M, W, t, a, b >$.
Output: Verification Success / Fail.
1: Generate $W_h = H(W||t)$.
2: Generate verification key $y \equiv (PK_A^{W_h} \times t) \in \mathbb{Z}_P^*$.
3: Calculate $k_v = (g^b \times y^a) \in \mathbb{Z}_P^*$.
4: Calculate $a_v = H(M||k_v) \in \mathbb{Z}_Q$.
5: **if** $(a == a_v)$ **then**
6: Success.
7: **else**
8: Fail.
9: **end if**

6 Anonymous Data Uploading Design

In this section, we present the data uploading mechanism of *Harpocrates*, so that data produced in a censoring network can become available to consumers outside of this network.

6.1 Evading the Censoring Network

*6.1.1 Data sharing initialization*The producer selects collaborating peers as a subset of the overall peers. The collaborating peers participate in uploading the producer's data outside of the censoring network. Subsequently, the producer creates uploading metadata for each collaborating peer, consisting of a symmetric key for the secure communication between the collaborating peer and the producer and the data pieces that the collaborating peer will forward to the proxies.

Fig. 4 illustrates a scenario, where the producer (peer A) distributes a subset of the total data to peer B, who will forward it to the proxies. The uploading metadata $[Uploading\_MetaData]_{PK_B}$ sent from peer A to B contains the symmetric key S_{AB} and a list of data pieces "`/sync/Game1/Piece_B_1`" $\cdots$ "`/sync/Game1/Piece_B_k`" that B should forward to the proxies. The metadata and the data pieces are named under a multicast synchronization ("sync" for short) prefix used by a multi-party application (*e.g.,* gaming) allowed to operate in the censoring network. This prefix masquerades the producer's prefix, so that it stays anonymous.

The producer sends uploading metadata to each of the collaborating peers (the metadata is encrypted using the receiving collaborating peer's public key) through the multicast synchronization channel. Thus, all peers in the multicast group will receive the metadata. However, only the peer with the corresponding private key (peer B in Fig. 4) will be able to decrypt the metadata and access the names of the data pieces that will be uploaded on behalf of the producer. In response, peer B will encrypt and send a decoy prefix (*e.g.,* "`/Mendeley`") back to A. Each data piece listed in the uploading metadata will contain one or more Data packets named by A under B's decoy prefix. Each of these packets will contain the data to be anonymously uploaded to the proxies. Once B receives a data piece, it will decapsulate the contained packets and forward them towards the proxies outside of the censoring network as we further discuss in Section 6.1.3.

*6.1.2 Data sharing modes among peers*Upon receiving the uploading metadata, a collaborating peer will request the data pieces specified in the metadata from the producer. We refer to this data sharing mode as *Pull*. These requests will be received by all peers in the multicast group, but only the producer has and will be able to provide the requested data.

Requests for data pieces may be intercepted by censoring nodes aiming to prevent the data from leaving the censoring network. The censoring nodes may receive requests sent by

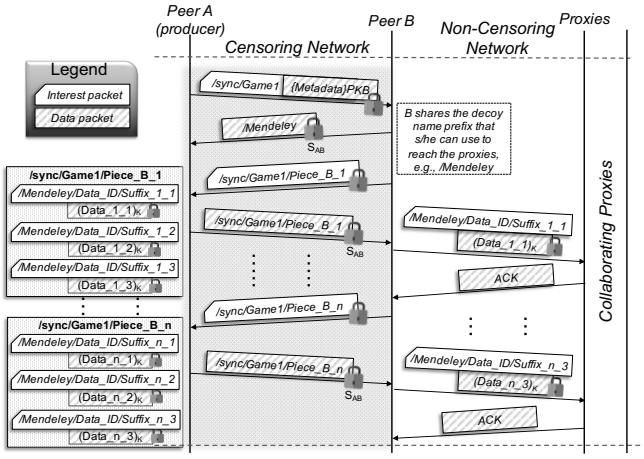


Figure 4: A data uploading example: peer A (producer) shares uploading metadata with peer B. Subsequently, peer B requests the data pieces specified in the metadata using the *Pull* communication mode. B forwards on behalf of the producer requests with decoy name prefixes contained in the received data pieces to collaborating proxies. These requests carry the data to be made available outside of the censoring network in an encrypted format.

collaborating peers and reply with bogus pieces. In the example of Fig. 4, B can detect a received bogus piece after trying to decrypt it using S_{AB} (shared symmetric key between A and B). As a result, B will request such pieces multiple times, alerting the producer that it has not received the legitimate pieces. Once the producer receives a certain number of consecutive requests for the same piece, the anti-censorship mode (*Push* data sharing mode) will be triggered. Under the *Push* mode, A will attach a piece requested multiple times onto an Interest and send ("push") it through the multicast channel to B (encrypted with B's public key).

Harpocrates features an adaptive communication mode (*Hybrid* data sharing mode) that operates under the *Pull* mode as long as no suspicious censorship activities are detected by the producer, while switching to the *Push* mode when *Harpocrates* detects censoring activities. This adaptation will happen by the producer independently for each collaborating peer, since censoring nodes may be closer to the producer (thus being able to block requests) than only certain collaborating peers. To this end, the producer maintains a status for each collaborating peer and monitors the delivery progress of corresponding pieces.

6.1.3 Making data available outside of the censoring network As illustrated in Fig. 4, once a collaborating peer receives and decrypts a data piece from the producer, this piece may contain one or more requests (Interests) for a decoy prefix. These requests carry ("hide") the data to be uploaded in an encrypted format. As we mentioned in Section 3.2, given the pull-based nature of NDN communication, where data can be retrieved only after the reception of a request, access to the censoring network from the outside world may be easily restricted by the censor. To evade censorship and make the data available outside of the censoring network, the collaborating peers send the requests, found in the received pieces, towards the proxies. Due to their decoy name prefixes, these requests will be forwarded outside of the censoring network.

6.2 Data Gathering and Reconciliation

As we explained in Section 5, the producer generates and shares with the selected proxy a *Data ID* random string. This is included in the names of the requests sent from the collaborating peers to the collaborating proxies and is used to signal the selected proxy that these requests carry data belonging to a particular data collection. The selected proxy shares the *Data ID* value with all collaborating proxies, instructing them to forward all the packets they receive and that contain this value in their names to the selected proxy. For instance, Fig. 4 illustrates that peers A and B agreed to use "`/Mendeley`" as the decoy prefix, while the requests sent to the collaborating proxies have a name prefix "`/Mendeley/Data_ID`". Collaborating proxies receiving Interests for "`/Mendeley`" followed by "`/Data_ID`" will forward them to the selected proxy. The suffix of the names can be selected by the producer based on the naming patterns of legitimate applications that use the decoy prefix, maximizing the resemblance between these requests and legitimate requests for the decoy prefix.

The requests received by the collaborating proxies carry the data to be uploaded in an encrypted format, however, the selected proxy is the only entity that can decrypt this data, since it possesses the symmetric key K shared by the producer during the secure delegation process (Fig. 3). As a result, only the selected proxy can gather all the data, decrypt it, and reconcile the original data collection generated by the producer. The reconciled data will be published by the selected proxy to consumers under the name instructed by the producer during the secure delegation process (Fig. 3).

7 Evaluation

In this section, we present our evaluation study under two setups. We first implement and evaluate our proxy signature design on different hardware platforms. We then implement *Harpocrates* and perform network simulations, so that we can scale our study to large network topologies. Finally, we compare *Harpocrates* to a design based on onion routing [17].

7.1 Evaluation Setup

To evaluate the security delegation phase (Section 5), we implemented the proxy signature [2] and Schnorr signature [34] mechanisms using the Charm-Crypto library [4]. We developed the proxy signature generation (Protocol 1), the proxy signing (Protocol 2), and the proxy verification (Protocol 3) protocols. We also implemented Schnorr message signing and signature verification as our comparison baseline. We benchmarked these protocols on three platforms: (i) a Raspberry Pi 4 with an ARMv7 processor and 4GB of RAM running Raspbian 10; (ii) a laptop with a 2.20GHz Intel Core-i7 processor and 4GB of RAM running an Ubuntu 16 Virtual Machine (VM); and (iii) a desktop class server with

a 3.60GHz Intel Xeon processor and 16GB of RAM running Ubuntu 18. The results are averaged over 500 runs.

We use ndnSIM [28], the de-facto NDN network simulator, to implement and evaluate *Harpocrates* based on a Rocketfuel topology (AS1221) with 278 routers and 731 links [37]. We connect collaborating peers and censoring nodes to this topology by creating links to randomly selected routers. We randomly attach five proxies to the topology, while ensuring that the distance between each proxy and the closest peer is at least five hops, so that each proxy is out of the censoring network. A file of size 100MB is generated by a producer (randomly selected among the peers) and is sent towards the proxies. Finally, we implemented a design based on onion routing [17] to compare with *Harpocrates*. The realization of such an onion-based routing design in NDN is a challenge on its own, since NDN is fundamentally different than TCP/IP. To this end, in this paper, we randomly selected three onion routers and incorporated benchmarked encryption/decryption times of onion encryption operations for each onion router. For simplicity, we did not consider the time for the selection of onion routers and key exchanges. The results are averaged over ten runs.

Evaluation metrics: We consider the following metrics:
(1) *Run time of proxy and Schnorr signing and verification*: the time needed to perform the signing and verification operations on different hardware platforms. Proxy signing includes the time for proxy key derivation and Schnorr signature. Similarly, the proxy verification run time includes the time for proxy key derivation and Schnorr signature verification.
(2) *Data distribution success rate*: the percentage of the total data that was successfully uploaded to the proxies.
(3) *Data publication delay*: the time elapsed between the producer generating the data and the completion of the reception of all the data by the proxies.
(4) *End-to-end per packet delay*: the time elapsed between starting the data uploading process for each Data packet and the reception of each packet by a proxy.
(5) *Normalized overhead*: the ratio between the volume of overhead traffic (multicast communication, metadata exchanges, peer-to-peer data sharing) and the volume of the data to be uploaded from the producer to the proxies. We further normalize the overhead based on the traffic volume generated by the *Pull* mode. To this end, the *Pull* mode will, by definition, result in normalized overheads of value 1.

7.2 Evaluation Results

Run time of proxy and Schnorr signing and verification: As shown in Fig. 5, the proxy signature generation, executed by the producer, does not incur considerable delay even when running on a constrained device (6ms on a Raspberry Pi). The proxy signing process results in run times of about 3× higher than the run times for Schnorr signature on all platforms. The additional cost is attributed to the generation of the corresponding public key (for signature verification) and matching this key against the producer's public key (Lines 1-3 of Protocol 2). Similarly, the proxy verification results in run times of about 1.5× higher than Schnorr signature verification on all platforms due to the proxy's public key derivation (Lines 1-2 of Protocol 3). We emphasize that the key derivation and comparison (Protocols 2 and 3) are executed only once per uploading session, incurring a negligible cost when amortized over multiple signing and signature verification operations.

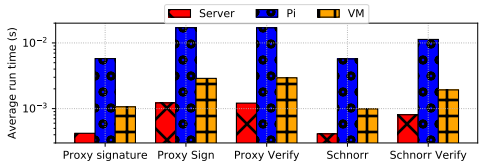


Figure 5: Proxy and Schnorr signature implementation across different platforms. Run times are shown in log-scale.

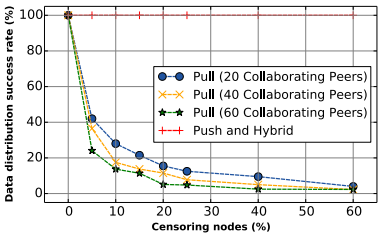


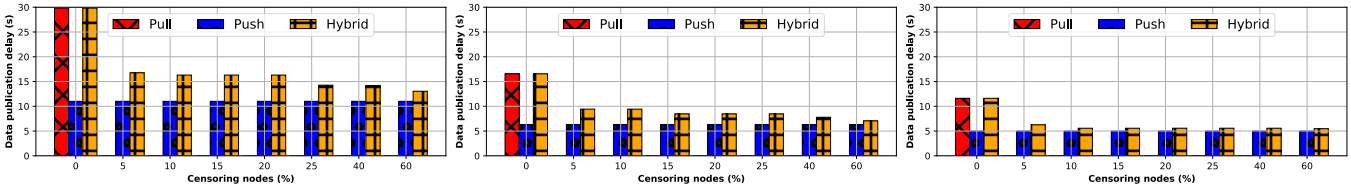
Figure 6: Success rate of different *Harpocrates* data sharing modes (*Pull*, *Push*, and *Hybrid*). *Push* and *Hybrid* are aggregated into a single line as their success rates overlap.

Table 2: Percent of collaborating peers blocked by censoring nodes.

Collaborating Peers	Censoring Nodes (%)							
	0%	**5%**	**10%**	**15%**	**20%**	**25%**	**40%**	**60%**
20	0%	58%	72%	78.5%	84.5%	87.5%	90.5%	96%
40	0%	63.5%	82.5%	86.25%	88.5%	92%	95%	97.75%
60	0%	75%	86%	88.5%	94.8%	95%	97.5%	97.6%

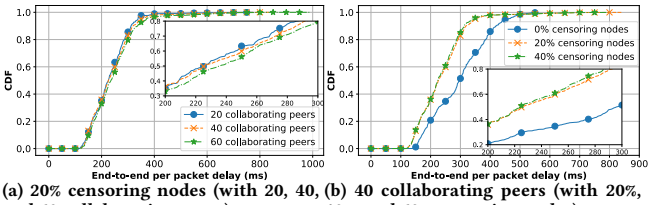
Data distribution success rate: In Fig. 6, we present the data distribution success rate. Our results show that *Hybrid* and *Push* modes successfully upload all the produced data to the proxies. On the other hand, in the case of *Pull*, censoring nodes are able to intercept the requests for data pieces sent by the collaborating peers towards the producer. To this end, collaborating peers will not be able to receive and distribute the data towards the proxies. The actual success rate values depend on the actual placement of the censoring nodes. However, the random placement in our experiments shows that even for small percentages of censoring nodes (5% to 10%), the success rate of *Pull* degrades considerably, since the majority of collaborating peers is blocked by censoring nodes as presented in Table 2. Specifically, 58-75% and 72-86% of the collaborating peers are blocked for 5% and 10% of censoring nodes respectively. As the percentage of censoring nodes increases, up-to 96-97.75% of the collaborating peers may be blocked. Nevertheless, even in such cases, *Harpocrates* successfully uploads all the produced data to the proxies.

Data publication delay: In Fig. 7, we present the results of the average data publication delay for *Pull*, *Push*, and *Hybrid*. Our results indicate that the data publication delay for *Push* is the lowest and it does not increase as the number of censoring nodes increases, since the data pieces are pushed to



(a) 20 collaborating peers (b) 40 collaborating peers (c) 60 collaborating peers

Figure 7: Data publication delay for varying numbers of collaborating peers and percentages of censoring nodes. Results for *Pull* are omitted when it fails to successfully make all the data available outside of the censoring network.



(a) 20% censoring nodes (with 20, 40, and 60 collaborating peers) (b) 40 collaborating peers (with 20%, 40%, and 60% censoring nodes)

Figure 8: CDF of the end-to-end per packet delay. Markers do not represent actual data points, but are only used for better readability.

all nodes including the censors, while only the collaborating peers can decrypt these pieces. *Pull*'s performance suffers in the presence of censoring nodes, even if their number is relatively small (*e.g.,* 5% or 10% of the number of collaborating peers). When the percentage of censoring nodes increases from 0% to 5%, *Pull* fails to distribute all data pieces among the collaborating peers. *Hybrid*, however, successfully adjusts to the censoring nodes that intercept the data pieces, switching to the *Push* mode. Our results show that as the percentage of censoring nodes increases, *Hybrid* switches from the *Pull* to the *Push* mode sooner during the data publication process, thus *Hybrid*'s data publication delay converges towards the delay of *Push*. For all the modes, the data publication delay decreases as we increase the number of collaborating peers due to the fact that more peers upload the data in parallel.

Further analysis of our results indicated that 3-8% of the data publication delays are spent on sharing the metadata between producers and peers, 47-56% on sharing the data pieces between producers and peers, and 36-50% on sending the actual data from the collaborating peers to the proxies. Note also that the *Hybrid* mode results in 1.5-2.1× higher publication delays than uploading the data from the producer to the closest proxy directly over the shortest network path.

End-to-end per packet delay: Fig. 8 presents the CDF of the per packet delay. Fig. 8a shows that for varying numbers of collaborating peers (same percentage of censoring nodes), 40% and 80% of the data is uploaded in less than 200ms and 300ms respectively. The per packet delay slightly increases with the number of collaborating peers, since these peers may be further away from the producer, thus the data pieces travel longer distances to reach them. Fig. 8b shows that the per packet delay decreases as the percentage of censoring nodes increases, since more pieces are blocked, thus *Harpocrates* switches from *Pull* to *Push* sooner during data publication.

Normalized overhead: Fig. 9 shows the normalized overhead results. The overhead for *Pull* is equal to 1, since it acts as the normalization factor. *Push* results in the highest overheads, since the data pieces are attached onto Interests pushed towards collaborating peers. *Hybrid* successfully copes with the interception of data pieces by censoring nodes, achieving overheads in the range between *Pull* and *Push*. It converges to the overhead of *Pull* when no or a few censoring nodes exist and to the overhead of *Push* as the number of censoring nodes increases. As the number of collaborating peers increases, the overhead for *Push* and *Hybrid* increases, since the size of the peer multicast group increases.

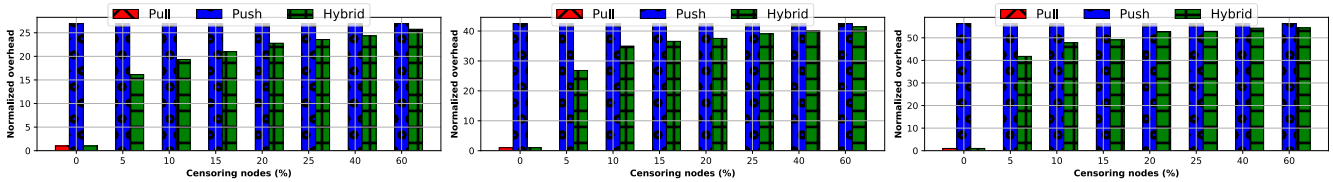
Comparison to an onion routing based design: Compared to a design based on onion routing, *Harpocrates* achieves 1.33-4.05× lower data publication delays, since it does not require multiple time-consuming layers of encryption/decryption. Depending on the placement of onion routers, *Harpocrates* incurs roughly the same to up to 1.51× lower overheads for *Pull* and *Hybrid* compared to the onion routing based design when no censoring nodes exist. As we increase the number of censoring nodes and collaborating peers, the *Hybrid* mode of *Harpocrates* incurs 1.21-2.05× higher overheads compared to the design based on onion routing, since: (i) it switches from *Pull* to *Push* sooner during data publication as we increase the number of censoring nodes; and (ii) the size of the peer multicast group increases as we increase the number of collaborating peers.

8 Security Analysis and Discussion

In this section, we discuss further security considerations and directions to extend the design of *Harpocrates*.

Censoring network nodes: A censoring authority may deploy censoring nodes in the network, including routers and Deep Packet Inspection (DPI) proxy firewalls, to interrupt data publication or breach the producer's anonymity. A censoring router, due to its limited capability in processing Interest and Data packets beyond name matching, can randomly drop a subset of packets. Such an action will negatively impact peers that legitimately use allowed communication channels. Prior work has argued that censoring authorities avoid actions that result in high collateral damage [48].

Malicious routers may redirect traffic portions to proxy firewalls for DPI. Such redirection in NDN is complicated due to the communication model symmetry. More importantly, Interest and Data packets, although semantically rich, do not carry fine-grained information that is available in TCP/IP packets (*e.g.,* IP addresses and port numbers). We argue that in inspecting NDN packets, a proxy firewall can



(a) 20 collaborating peers (b) 40 collaborating peers (c) 60 collaborating peers

Figure 9: Normalized overhead for varying numbers of collaborating peers and percentages of censoring nodes. Results for *Pull* are omitted when it fails to successfully make all the data available outside of the censoring network.

only use packet sizes, names, and signature related information. *Harpocrates* limits the impact of these threats by: (i) uploading small-sized Data packets by attaching them to Interests to avoid traffic anomalies; (ii) revealing innocuous names (*i.e.,* used by traffic allowed in the censoring network); and (iii) referring to anonymous certificates in the signature related information [39] of data pieces to prevent the producer's linkability to the data. It will be computationally expensive for a censoring authority to verify the signatures of all data pieces. However, if DPI drops all pieces associated with anonymous certificates, *Harpocrates* will switch from the *Pull* to the *Push* mode, piggybacking pieces onto Interests (typically not signed in most NDN applications).

Censoring peers: The censoring authority may deploy censoring nodes among peers. A censoring peer may intercept requests for data pieces from collaborating peers and reply with bogus pieces, which will consume PIT entries on routers and prevent the legitimate pieces from reaching the collaborating peers. The *Hybrid* mode of *Harpocrates* thwarts this threat by switching to *Push* when such an event is identified. As we discussed in Section 7, the *Hybrid* mode achieves 100% data distribution success rates in the presence of censoring peers–even when 60% of the peers are malicious.

If censoring nodes are among the collaborating peers, these censoring collaborating peers can interrupt the communication by obtaining and dropping the producer's data pieces (blackhole attacks). Although we assumed that the collaborating peers are not malicious (Section 3.2), here we discuss directions to thwart such an attack. The first direction involves data replication. In a naive approach, the producer blindly replicates the data by communicating overlapping data portions to different collaborating peers. This increases the chances that the data will be received by legitimate collaborating peers, who will upload it towards the proxies. To minimize redundant data delivery, the producer can obtain the list of missing Data packets from the selected proxy and publish them through the collaborating peers that delivered previous packets. The producer identifies legitimate collaborating peers by tracking their success rates in delivering data to the selected proxy. Network coding techniques, such as Random Linear Network Coding [19], can also be employed to deliver linearly independent combinations of Data packets to the selected proxy, enabling efficient data reconciliation.

The second direction involves group-oriented cryptographic techniques such as attribute-based [18] and broadcast [14] encryption. These techniques enable a group of collaborating peers to use their private keys to independently decrypt the same data piece delivered to them during the *Push* mode over the multicast communication channel. If, at least, one of the collaborating peers that can decrypt each data piece is legitimate, the data will be successfully uploaded to a proxy.

Producer and collaborating peer anonymity: In our design, the producer includes its public key in the warrant, enabling consumers to verify the validity of the delegation in addition to the proxy's signatures. The producer's public key in the warrant may allow the censoring authority to identify the producer, compromising its anonymity. To cope with this threat, approaches that provide signature anonymity can be used, including attribute-based [32], ring [33], and group signatures [9]. The producer's anonymity can be also augmented through a transient key cryptosystem [8], an asymmetric key cryptosystem, in which the key pair is bound to a short time period rather than the owner's identity. Thus, a singed Data packet will be associated with a time (delegation initiation in *Harpocrates*) rather than an identity. However, utilizing such a cryptosystem requires further considerations since private keys will be deleted after their short expiry time.

A malicious producer (deployed by the censoring authority) may be able to infer the participation of collaborating peers in data uploading, compromising their privacy. Similar to the producer's anonymity, cryptosystems including ring, group, and attribute-based signatures can preserve peers' anonymity. Distributed anonymous reputation management mechanisms can also help peers make informed decisions about their participation in data uploading [43].

Traffic analysis attacks: The censoring authority may orchestrate traffic analysis attacks to infer communication patterns from encrypted traffic, aiming to breach the producer's anonymity. Note that data producers in *Harpocrates* use legitimate communication channels to transfer their data to the collaborating peers and subsequently to the proxies. Leveraging such legitimate communication channels for distributing the data between the collaborating peers hides the producer's data and prevents the censoring authority from identifying a data upload attempt. As described in Section 6.1, dispersing the producer's data across multiple collaborating peers allows each peer to obtain a small portion of the data–with potentially different sizes–from the producer. Peers can send requests for data to the producer such that the generated traffic follows the legitimate application distribution, making these requests indistinguishable from the traffic generated by the legitimate application. Each peer also uploads a small

portion of the generated data to the proxies, thus preventing peers from sending abnormal amounts of data outside of the censoring network and creating traffic anomalies.

To transfer the producer's data to proxies outside of the censoring network, the collaborating peers send requests containing portions of the data generated by the producer hidden in them. The censoring authority may attempt to orchestrate traffic correlation attacks by passively observing the packet sizes between the producer and the collaborating peers or between the collaborating peers and the proxies. However, the data producer can assign different portions of the data to collaborating peers, ensuring that traffic patterns between the producer and these peers are not identical. NDN also features variable size request packets, since such packets can carry an unbounded number of parameters (data of arbitrary sizes) as defined by the NDN packet format [1]. As a result, data producers can generate Interests of variable sizes that follow the packet sizes of legitimate applications.

9 Conclusion and Future Work

In this paper, we presented *Harpocrates*, a framework for the anonymous publication of data from a censoring network to users outside of this network. *Harpocrates* takes advantage of communication channels and applications that are allowed in the censoring network, maximizing the collateral damage for censoring authorities. By employing different data sharing modes, *Harpocrates* can defend against censoring actions. Through a secure delegation mechanism, *Harpocrates* enables proxies outside of a censoring network to make data available to users without compromising the producer's anonymity. In the future, we plan to: (i) implement a *Harpocrates* prototype and evaluate it against other censorship circumvention solutions; and (ii) design mechanisms to defend against malicious collaborating peers and proxies.

Acknowledgements

This work is partially supported by National Science Foundation awards CNS-2104700, CNS-2016714, and CBET-2124918, the National Institutes of Health (NIGMS/P20GM109090), the Nebraska University Collaboration Initiative, the Nebraska Tobacco Settlement Biomedical Research Development Funds, and Intel Labs through a gift.

References

[1] [n.d.]. NDN Packet Format Specification-Interest Parameters. https://named-data.net/doc/NDN-packet-spec/current/interest.html#applicationparameters.

[2] S. Aboud and S. Yousef. 2012. A practical proxy signature scheme. *International Journal of Digital Information and Wireless Communications* 2, 4 (2012), 296–305.

[3] Bengt Ahlgren et al. 2012. A survey of information-centric networking. *IEEE Communications Magazine* 50, 7 (2012), 26–36.

[4] Joseph A. Akinyele et al. 2013. Charm: a framework for rapidly prototyping cryptosystems. *Journal of Cryptographic Engineering* (2013). https://doi.org/10.1007/s13389-013-0057-3

[5] S. Arianfar, T. Koponen, B. Raghavan, and S. Shenker. 2011. On preserving privacy in content-oriented networks. In *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*. ACM.

[6] C. Bernardini, S. Marchal, M. Asghar, and B. Crispo. 2019. PrivICN: Privacy-preserving content retrieval in information-centric networking. *Computer Networks* 149 (2019), 13–28.

[7] Cecylia Bocovich and Ian Goldberg. 2016. Slitheen: Perfectly imitated decoy routing through traffic replacement. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security*. ACM.

[8] Gilles Brassard. 1983. Relativized cryptography. *IEEE Transactions on Information Theory* 29, 6 (1983), 877–894.

[9] David Chaum and Eugène Van Heyst. 1991. Group signatures. In *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer, 257–265.

[10] S. DiBenedetto et al. 2011. ANDaNA: Anonymous Named Data Networking Application. *Arxiv preprint arXiv:1112.2205* (2011).

[11] Zakir Durumeric, Eric Wustrow, and J Alex Halderman. 2013. ZMap: Fast Internet-wide scanning and its security applications. In *Presented as part of the 22nd {USENIX} Security Symposium*.

[12] Tariq Elahi, Kevin Bauer, Mashael AlSabah, Roger Dingledine, and Ian Goldberg. 2012. Changing of the guards: A framework for understanding and improving entry guard selection in Tor. In *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*. 43–54.

[13] Roya Ensafi, Philipp Winter, Abdullah Mueen, and Jedidiah R Crandall. 2015. Analyzing the Great Firewall of China over space and time. *Proceedings on privacy enhancing technologies* 2015, 1 (2015), 61–76.

[14] Amos Fiat and Moni Naor. 1993. Broadcast encryption. In *Annual International Cryptology Conference*. Springer, 480–491.

[15] David Fifield et al. 2015. Blocking-resistant communication through domain fronting. *Proceedings on Privacy Enhancing Technologies* (2015).

[16] N. Fotiou, D. Trossen, G. F. Marias, A. Kostopoulos, and G. C. Polyzos. 2014. Enhancing information lookup privacy through homomorphic encryption. *Security and Communication Networks* (2014).

[17] David Goldschlag, Michael Reed, and Paul Syverson. 1999. Onion routing. *Commun. ACM* 42, 2 (1999), 39–41.

[18] Vipul Goyal et al. 2006. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM conference on Computer and communications security*.

[19] Tracey Ho et al. 2003. The benefits of coding over routing in a randomized setting. (2003).

[20] Dijiang Huang. 2010. Anonymous certification services. In *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*. IEEE.

[21] J. Karlin et al. 2011. Decoy Routing: Toward Unblockable Internet Communication. In *USENIX Workshop on Free and Open Communications on the Internet FOCI*. USENIX Association.

[22] Josh Karlin, Daniel Ellard, Alden W Jackson, Christine E Jones, Greg Lauer, David Mankins, and W Timothy Strayer. 2011. Decoy Routing: Toward Unblockable Internet Communication.. In *FOCI*.

[23] Kentaro Kita et al. 2020. Producer Anonymity based on Onion Routing in Named Data Networking. *IEEE Transactions on Network and Service Management* (2020).

[24] Christopher S Leberknight, Mung Chiang, Harold Vincent Poor, and Felix Wong. 2010. A taxonomy of Internet censorship and anti-censorship. In *Fifth International Conference on Fun with Algorithms*.

[25] N. Leshov, M. A. Yaqub, M. Khan, S. Lee, and D. Kim. 2019. Content Name Privacy in Tactical Named Data Networking. In *Eleventh International Conference on Ubiquitous and Future Networks*. IEEE.

[26] Tianxiang Li, Wentao Shang, Alex Afanasyev, Lan Wang, and Lixia Zhang. 2018. A brief introduction to ndn dataset synchronization (ndn sync). In *2018 IEEE Military Communications Conference*. IEEE.

[27] M. Mambo, K. Usuda, and E. Okamoto. 1996. Proxy signatures: Delegation of the power to sign messages. *IEICE transactions on fundamentals of electronics, communications and computer sciences* (1996).

[28] Spyridon Mastorakis et al. 2017. On the evolution of ndnSIM: An open-source simulator for NDN experimentation. *ACM SIGCOMM Computer Communication Review* 47, 3 (2017), 19–33.

[29] H. Mozaffari, A. Houmansadr, and A. Venkataramani. 2019. Blocking-Resilient Communications in Information-Centric Networks using Router Redirection. In *Globecom Workshops (GC Wkshps)*. IEEE.

[30] Milad Nasr et al. 2017. The waterfall of liberty: Decoy routing circumvention that resists routing attacks. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. ACM.

[31] Milad Nasr et al. 2019. Enemy At the Gateways: Censorship-Resilient Proxy Distribution Using Game Theory.. In *NDSS*.

[32] Sanjeev Kaushik Ramani et al. 2019. NDN-ABS: Attribute-Based Signature Scheme for Named Data Networking. In *Proceedings of the 6th ACM Conference on Information-Centric Networking*. 123–133.

[33] Ronald L Rivest, Adi Shamir, and Yael Tauman. 2001. How to leak a secret. In *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 552–565.

[34] C.-P. Schnorr. 1991. Efficient Signature generation by Smart Cards. *Journal of Cryptology* 4, 3 (1991), 161–174.

[35] Max Schuchard, John Geddes, Christopher Thompson, and Nicholas Hopper. 2012. Routing around decoys. In *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 85–96.

[36] Irina Shklovski et al. 2011. Online contribution practices in countries that engage in internet blocking and censorship. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*.

[37] Neil Spring et al. 2002. Measuring ISP topologies with Rocketfuel. *ACM SIGCOMM Computer Communication Review* (2002).

[38] F. Tao et al. 2015. Secure Network Coding-Based Named Data Network Mutual Anonymity communication Protocol. In *Proceedings of International Conference on Electrical, Computer Engineering and Electronics*.

[39] NDN Team. 2020. NDN packet specification (signature field). https://named-data.net/doc/NDN-packet-spec/current/signature.html#data-signature

[40] Tor [n.d.]. Tor Project: Anonymity Online. http://www.torproject.org/.

[41] R. Tourani et al. 2015. Catch Me If You Can: A Practical Framework to Evade Censorship in Information-Centric Networks. In *Proceedings of the ACM Conference on Information-Centric Networking*. ACM.

[42] R. Tourani, S. Misra, T. Mick, and G. Panwar. 2018. Security, Privacy, and Access Control in Information-Centric Networking: A Survey. *IEEE Communications Surveys Tutorials* 20, 1 (2018), 566–600.

[43] Xinlei Oscar Wang et al. 2013. Artsense: Anonymous reputation and trust in participatory sensing. In *Proceedings of IEEE International Conference on Computer Communications*. IEEE, 2517–2525.

[44] Philipp Winter et al. 2012. *How the great firewall of china is blocking tor*. USENIX-The Advanced Computing Systems Association.

[45] Philipp Winter, Roya Ensafi, Karsten Loesing, and Nick Feamster. 2016. Identifying and characterizing Sybils in the Tor network. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*. 1169–1185.

[46] Eric Wustrow et al. 2011. Telex: Anticensorship in the Network Infrastructure.. In *USENIX Security Symposium*.

[47] Lixia Zhang et al. 2014. Named data networking. *ACM SIGCOMM Computer Communication Review* 44, 3 (2014).

[48] Hadi Zolfaghari and Amir Houmansadr. 2016. Practical censorship evasion leveraging content delivery networks. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*.