AND: High Performance Anonymous Named Data

Christopher A. Wood*
University of California Irvine
Irvine, CA, USA
woodc1@uci.edu

Gene Tsudik
University of California Irvine
Irvine, CA, USA
gts@ics.uci.edu

Ersin Uzun
PARC
Palo Alto, CA, USA
Ersin.Uzun@parm.com

ABSTRACT

With the increasing trend of personal and private activities and correspondence conducted on the Internet, the need for anonymous communication provided by services such as Tor continues to escalate. Furthermore, as information-centric networking (ICN) gains momentum, the need for a Tor-like analog to support high performance anonymous communication rises at an equal rate. ICN designs such as Named Data Networking are unique in that all communication is content-based rather than host-based, i.e., content, rather than hosts, are addressable in the network. In order to mask user locations and identities, it becomes necessary to mask the process in which content is requested. ANDaNA (Anonymous Named Data Networking Application) was the first attempt to provide this type of anonymity in the context of NDN. However, its elementary design and implementation led to severe performance issues that would hinder widespread usage. To this end, we propose AND, a new and vastly improved variant of ANDaNA, to support high throughput and low latency anonymous content retrieval in NDN in unidirectional and bidirectional settings. In this work, we describe the design, implementation, and performance of AND, and also prove various anonymity guarantees provided by our design. Our experimental results show that AND introduces minimal overhead with a minimal number of anonymizing proxies (1), while simultaneously ensuring anonymity guarantees similar to Tor.

Categories and Subject Descriptors

H.4 [**TODO**]: TODO; D.2.8 [**TODO**]: TODO—*TODO*

General Terms

NDN, Anonymity, Tor

1. INTRODUCTION

*NSF GRFP blurb.

Usage of the Internet has undergone a tremendous transformation since its inception in the 1970s. Content distribution, as opposed to point-to-point communication, has become the leading type of traffic traversing today's valuable network resources; Netflix, for example, accounted for nearly 30% of all downstream traffic in 2012 [1]. The number and popularity of such information-centric services are only expected to increase in the future with the growing presence of data-intensive consumer applications and devices (e.g., media streaming applications and mobile devices), leading to added pressure on network resources and a subsequent increase in network congestion and wasted bandwidth.

Named-data networking (NDN) [8] is an emerging network architecture capable of supporting information-centric traffic. Two primary characteristics of the NDN architecture are that content names, rather than hosts or locations, are addressible and routable through the network, and all generated content corresponding to some name must be signed by its original producer. The latter property decouples content confidentiality, integrity, and authenticity from content and the manner in which it is delivered to consumers (e.g., instead of using secure tunnels akin to SSH/TLS, content can be encrypted before sent to a consumer). These fundamental design decisions enable content to be cached in network-layer resources throughout the network, thus promoting reduced network congestion and wasted bandwidth when popular content is requested.

The NDN design builds content security support *into the architecture* and promotes content access via producer-specified forms of encryption. Similar to the IP-based counterpart, consumer and producer anonymity, however, are not readily supported by this design. While there are no longer source or destination addresses associated with traffic, there are a variety of other sources of information via which consumers can be deanonymized, including: interest and content names, network router cache contents, and digital signature contents. An intelligent adversary may use any of these information sources when launching client deanonymization attacks.

ANDāNA, an anonymous named-data networking application, pioneered support for anonymous content retrieval and distribution with regards to consumers and producers, respectively [2]. Inspired by Tor [3], ANDāNA uses onion-encryption to wrap requests for content that are iteratively decrypted and forwarded by participating anonymizing routers,

and also to wrap content as it flows from the producer to the consumer. Unlike Tor, however, ANDāNA was a proof-of-concept application-layer anonymizing layer for NDN that targeted unidirectional traffic *without* real-time requirements. Support for high-throughput, low-latency, bidirectional traffic, such as traffic generated from voice communication, video chat, and media streaming applications. In addition to this severe performance impediment, the "optimized" session-based variant of ANDāNA sacrificed interest and content linkability (an anonymity issue) for only slightly improved efficiency.

Consequently, in this work we present a new and substantially improved design for an NDN anonymizng application, henceforth referred to as AND, that addresses the many performance and anonymity pitfalls of the original design. We discuss the design and implementation of AND as built upon CCNx [6], the open-source relative of NDN. We also present performance results from testing our application implementation in numerous test environments with different types of traffic. Our results show that our modified design leads to substantial performance gains without sacrificing consumer or producer privacy, thus making AND a promising application to support a diverse set of application content and use cases in future information-centric networks.

2. PRELIMINARIES

In this section we give a more detailed overview of the properties of the NDN architecture relevant to the issue of anonymous communication. We then also delve into the design of ANDāNA (henceforth referred to as ANDāNA), to identify the major security flaws and engineering shortcomings that are remedied in ANDāNA-v2.

2.1 NDN Overview

Named-data networking (NDN) is one of several proposed information-centric network designs under active research and development as the future Internet architecture. Its defining characteristic is that it decouples the location of data from its original publisher. All requested content is cached in network-layer routers between consumers who request data and publishers, or other routers, satisfying such data. In essence, network caches and addressable content, rather than addressable hosts or interfaces, enable this new architecture to reduce network congestion and latency by keeping content closer to its intended recipients.

Content is requested through the issuance of an *interest*, or a meaningful URI with a one-to-one correspondence with content provided by the network. The components of an interest are arbitrary strings and can therefore be used to store any type of data, including human readable names or binary data encoded as URI-friendly strings. Upon receiving an interest, a router looks for a match in its *content store* (CS), which is the cache that stores content already requested from other consumers. Matching is done using exact-match based on content names, i.e., a complete interest name match in the CS will cause the associated content to be forwarded downstream on the same router interface upon which the interest arrived. Interests that do not completely match any content name in the CS are stored in a *pending interest table* (PIT) together with the corresponding interface upon which the interest arrived and is subsequently forwarded to the appropriate upstream router based on contents in a *forward interest base* (FIB) table. Multiple interests matching the same name are collapsed into a single PIT entry to prevent redundant interests being sent upstream. Once a content matching a PIT entry is received by a router, the content is cached (unless the interest has explicitly marked the content to not be stored) and sent to all downstream interfaces associated with the PIT entry. Finally, upon completion, the PIT entry is cleared.

Content-centric traffic also has strong security implications. Firstly, it means that content security is tied to the data itself, not the channel through which it flows between consumers and the network. All sensitive content must therefore be protected with a suitable form of encryption to ensure confidentiality. Content integrity is guaranteed by digital signatures; all content producers are required to sign content before responding to incoming interests. Due to performance impediments, signature verification is not required by network routers; naturally, consumer applications are expected to verify content signatures. Issues regarding signature verification and key management are discussed more at length in [].Secondly, a lack of source and destination addresses improves user privacy. However, as we will discuss in the following section, this lack of information is insufficient with regards to consumer privacy.

2.2 Notions of Anonymity

In the context of NDN there are many sources of information from which a user's privacy and anonymity can be compromised, including: (1) interest and content names, router cache contents, and content signatures. Anonymity in the context of NDN is defined with respect to both consumers $u \in \mathsf{C}$ (the set of all consumers) and producers $p \in \mathsf{P}$ (the set of all producers) - see Table 1 for a complete description of the notation used in this work. To motivate our design decisions (discussed in Section 3), we now define the security of our design in terms of consumer and producer anonymity and unlinkability. These can be found in [2], but we provide them here for completeness. We also provide new definitions that make sense in the context of the bidirectional session case that AND is intended to support.

Definition 1. *[2] For $u \in (\mathsf{C} \setminus \mathsf{C}_{\mathcal{A}})$, u is said to have* consumer anonymity *in configuration C with respect to adversary $\mathcal{A}$ if there exists $C' \equiv_{\mathcal{A}} C$ such that $C'(u') = C(u)$ and $u' \neq u$.*

Definition 2. *[2] Given $\overline{\mathsf{int}}_1^n$ and $p \in \mathsf{P}$, $u \in \mathsf{C}$ has* producer anonymity *in configuration C with respect to p and adversary $\mathcal{A}$ if there exists a configuration $C' \equiv_{\mathcal{A}} C$ such that $\overline{\mathsf{int}}_1^n$ is sent by a non-compromised consumer to a producer different from p.*

Definition 3. *Two entities u_1 and u_2, both serving as producer and consumer of content in an application session, are said to have* session anonymity *in configuration C with respect to adversary $\mathcal{A}$ if both u_1 and u_2 enjoy producer and consumer anonymity in C with respect to $\mathcal{A}$.*

Definition 4. *[2] We say that $u \in (\mathsf{C} \setminus \mathsf{C}_{\mathcal{A}})$ and $p \in \mathsf{P}$ are* unlinkable *in C with respect to an adversary $\mathcal{A}$ if there*

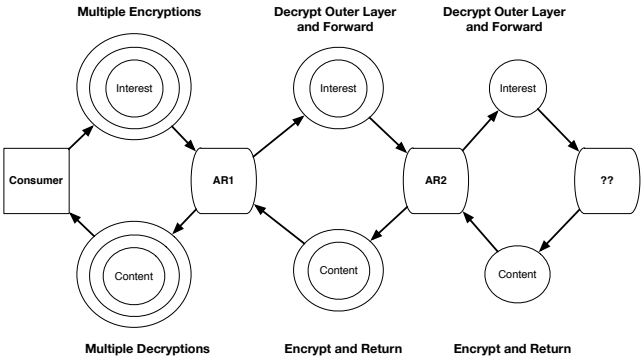

Figure 1: Interest and content onion encryption and decryption in the original ANDāNA design.

exists a configuration $C' \equiv_{\mathcal{A}} C$ *where* u*'s interests are sent to a producer* $p' \neq p$*.*

Our primary goal is to achieve consumer and producer anonymity and unlinkability with minimal network-layer overhead. We describe the notable design elements and prove of AND used to achieve this goal in Section 3, and prove claims of anonymity and unlinkability in Section 5.

2.3 ANDāNA Design Highlights

Section 3 provides a detailed description of the design of AND, which is inspired by Tor and its predecessor, ANDāNA. To motivate our changes we first describe the ANDāNA design in sufficient detail. As previously mentioned, ANDāNA achieves anonymous communication using mix behavior analogous to Tor; Interests for content are wrapped in layers of encryption, where each encrypted layer has sufficient information necessary to forward the interest to the correct anonymizing router. Note that unlike Tor, persistent circuits with long-term TCP connections are not established with ANDāNA. Rather, ephemeral circuits are created as an interest is decrypted and forwarded upstream towards the consumer. State information in each anonymizing router is only maintained in the symmetric (session-based) variant of ANDāNA. Once a piece of content is retrieved corresponding to an interest, each anonymizing router wraps it with a layer of encrypting using a public or symmetric key specific in the corresponding interest, and is subsequently sent back downstream. In this way, circuits only exist for the duration of a single interest-content retrieval. This behavior is illustrated in Figure 2.3.

In the symmetric variant of ANDāNA, state information consisting of a unique session identifier and symmetric key used for interest and content decryption and encryption, respectively, is established and persisted in each anonymizing router using a standard three-way handshake protocol. While the use of symmetric encryption removes the computational burden of asymmetric content encryption, the reference design required that the session identifier be sent in the clear for every interest, allowing an adversary to link packets and content together and render the consumer susceptible to a deanonymization attack (see the following section for more details). Furthermore, not only is the initial state establishment protocol composed of interests that are distinct from regular encrypted interests, the handshake procedure wastes consumer bandwidth and time when used for short-term anonymous traffic.

2.4 Design and Engineering Pitfalls

The primary motivation for a new design of ANDāNA is to attain the same anonymity and privacy guarantees as ANDāNA with *better* performance. The original design targeted a single use case in which performance, especially in the bidirectional setting, was not a primary concern. Indeed, there was both an asymmetric and symmetric (session-based) variant of ANDāNA, and while the latter enjoyed better speedups over the former by using symmetric encryption it suffered the fatal flaw of not ensuring packet unlinkability. It is generally the case that unlinkability is merely sufficient for anonymity, rather than also being a necessary condition for anonymity. However, in the case of ANDāNA, packet linkability can lead to consumer and producer linkability, which immediately violates anonymity. For example, it is not difficult to hypothesize an adversary that eavesdrops on incoming and outgoing interests for a particular anonymous router, and who by doing so is able to determine that the incoming and outgoing session IDs are linked. In fact, a modified type of this kind of adversary was explicitly studied in the context of Tor by Murdoch and Danezis in [5]. In their work, the goal of the adversary in their "linkability attack" was to determine whether two separate data streams being served by two corrupted servers were initiated by the same consumer, and we suspect that such analysis could be augmented to work for ANDāNA. Specifically, repeating a packet linkability attack at each anonymous router in a circuit may therefore eventually lead to linkability between the producer and consumer.

The use of application and environment contextual information has also been formally studied in [4], in which side channel and environment information (e.g., the deterministic behavior of an anonymous router always forwarding a packet upstead after unwrapping an interest received from some downstream router) is used to quantify the *degree of unlinkability*. Furthermore, we remark that regardless of how such linkability information is acquired, it has been shown that it can lead to reduce consumer and producer anonymity beyond what is possible with general traffic analysis [7].

As most of the literature focuses on mix-based anonymizing services akin to Tor, which inspired the original design of ANDāNA, it is clear that any form of linkability should be avoided in order to maintain consumer and producer anonymity. Therefore, one formal goal for AND is to attain the same anonymity and privacy guarantees as the asymmetric variant of ANDāNA, which does not suffer from packet linkability issues, while supporting *superior* performance in the low-latency, high-throughput, bidirectional traffic use case when compared to the symmetric variant of ANDāNA.

3. AND DESIGN

In this section we describe the much improved design of AND. Table 1 introduces the relevant notation needed to understand the design in its entirety.

Table 1: Notation used in the presentation of this work.

C	Set of all consumers
P	Set of all producers
R	Set of all routers
IF	Set of all interfaces on all routers
$\mathsf{if}_i^r \in \mathsf{IF}$	Interface i of router r
κ	The global security paramter (set to 256 in practice)
(pk_i, sk_i)	Public/private key pair for router r
$\overline{\mathsf{int}}_i^j$	Encrypted interest wrapped from router i to router j $(i \leq j)$
$\mathcal{A}$	Adversary
u	Entity in the network (consumer or producer)
$u \rightarrow_{\mathsf{int}} r$	Entity u sends interest to router r
$\mathsf{int} \rightarrow \mathsf{if}_i^r$	Interest int is sent to interface i of router r
$r \in \mathsf{R}$	An anonymizing router (AR)
$\mathcal{E}_{pk_i}(\cdot)$	Public key encryption using pk_i
$\mathcal{D}_{pk_i}(\cdot)$	Public key decryption using pk_i
$\mathsf{Encrypt}_{k_i}(\cdot)$	Symmetric key encryption using key k_i
$\mathsf{Decrypt}_{k_i}(\cdot)$	Symmetric key decrypt using key k_i
ST	AR session table used to store session ID and digest tuples
H	A collision resistant hash function on the domain $\{0,1\}^*$ to range $\{0,1\}^\kappa$
F_k	A keyed pseudorandom function

3.1 Circuit and Session Establishment

At the heart of the AND is the notion of anonymizing routers and connection-oriented circuits, similar in spirit to the inner workings of TOR [3]. Anonymizing routers serve two purposes in AND: (1) to decapsulate and forward encrypted interests, along with content encryption keys, generated by a consumer until the cleartext interest arrives at the producer, and (2) to encapsulate sensitive content using the previously acquired encryption keys and relay the encrypted content downstream. In this way, the consumer generates an interest wrapped by several layers of encryption and receives a piece of content wrapped in several layers of encryption that it can easily decrypt. It is also important to note that since each anonymizing router operates at the application layer, it effectively serves as the producer for each downstream router in the AND circuit. Therefore, NDN policy dictates that such content *must be signed*. Verification of content from upstream routers, however, is not mandatory.

We also note that the current AND design has support for two types of circuits: asymmetric and symmetric session-based. In the asymmetric variant, all encrypted interests are done using a CCA-secure PKI scheme and all content is encrypted using a CCA-secure symmetric key encryption scheme. Conversely, in the session-based variant, all encrypted interests are protected using a CCA-secure symmetric key encryption scheme, where the key is identified using a unique session identifier sent in the cleartext along with the encrypted interests. This worsens anonymity because it provides a way to link packets to a single session (as previously discussed).

Putting together all of the design aspects of the current version of AND, we see that the following factors weigh in on the overall performance of the application: content encryption, content signature generation, and encrypted interest generation. In AND we seek to minimize the degree to which these factors affect interest and content encapsulation and decapsulation by integrating support for anonymous router *state*, which is encapsulated in sessions, into anonymous circuits. Sessions will exist for unidirectional traffic only, which therefore means that bidirection traffic, the ultimate focus of this work, will require two sessions to be established and maintained for the duration of the bidirectional application. This is done so that each party in the application need not use the same set of anonymous routers for communication. Not only does this free each consumer to select a random subset of anonymous routers $r_1, r_2, \ldots, r_n$ from the set of total anonymous routers R, but it may also help improve QoS guarantees by distributing the load of encapsulation and decapsulation among multiple nodes. Furthermore, sessions enable the establishment of long-term secrets that can be used to improve the efficiency of certain cryptographic operations, such as content encryption, signature generation, and signature verification.

With the end of goal of supporting highly efficient and anonymous bidirectional traffic, the goals of the circuit and session establishment using a list of n anonymous routers $r_1, \ldots, r_n$ are as follows:

1. Establish unique session IDs $\mathsf{session}_i$ and session IVs SIV_i
2. Establish content encryption keys E_{k_i} and initial counter values EIV_i
3. Establish pairwise MAC keys M_{k_i} between adjacent routers and the consumer used to tag and verify content

The purpose of each of these session entities will become clear from the circuit initialization and usage procedures. After establishment, the circuit from the consumer C to the producer P should be similar to that shown in Figure 3. We now describe the general protocol for establishing this type of session-based circuit from a consumer C to a producer P given n anonymous routers. Recall that, in order to support

bidirectional communication, P would have to establish a similar circuit to C with m routers, where m need not equal n.

AND supports state initialization that is separate in time from circuit usage (i.e., using a handshake routine to initialize the state) as well as one that initializes session state upon issuance of the first wrapped consumer interest. In what follows we describe the handshake variant of session establishment. Online session establishment, used to remove the seemingly unnecessary handshake phase altogether, is discussed in Section 3.3.

Notice that by this procedure, no two routers will share the same session identifier (with non-negligible probability) even though they partake in the same circuit. This is because they generate session identifiers independent and uniformly at random from $\{0,1\}^\kappa$.

3.2 AND Circuit Usage

We note that anonymous routers must be chosen in a *mutually excluse* manner, meaning that they do not change share the same name prefix or are from the same organization (see Figure 2). This requirement is needed for ensuring anonymity. After a circuit and the corresponding sessions have been created between the consumer and each anonymous router, usage of the circuit proceeds as per the original AND design. Specifically, there are three main operations that need to be defined: encrypted interest generation, AR interest forwarding, and AR content handling. In what we follows we present the details of each of these procedures as needed for AND. We begin with the encrypted interest generation procedure (shown in Algorithm 2) in which a consumer C particpating in a particular *application* session with a producer P wraps an interest for the session to be sent into the anonymizing circuit. A wrapped (encrypted) interest from r_i to r_j $(i \leq j)$ is denoted as $\overline{\mathsf{int}}_i^j$, meaning that the original plaintext interests int cannot be retrieve unless encrypted by each router $r_i, r_{i+1}, \ldots, r_j$, in that order. Thus, the original wrapped interest is denoted as $\overline{\mathsf{int}}_1^n$.

Algorithm 2 Encrypted Interest Generation

Require: Interest int, circuit length n, AR pool $\mathcal{R}$
Ensure: Encrypted interest $\overline{\mathsf{int}}_1^n$

$\overline{\mathsf{int}} = \mathsf{int}$
for $i = n$ **downto** 1 **do**
  $\mathsf{SIndex}_i := H(\mathsf{session}_i + \mathsf{SIV}_i)$
  $\mathsf{SIV}_i = \mathsf{SIV}_i + 1 \pmod{2^\kappa}$
  $\overline{\mathsf{int}}_i^n = R_i / \mathsf{SIndex}_i / \mathsf{Encrypt}_{E_{k_i}}(\overline{\mathsf{int}}, \mathsf{timestamp})$
return $\overline{\mathsf{int}}_1^n$

3.3 Online Session Establishment and Circuit Usage

As previously mentioned, sessions may be created via the handshake technique or online. The process for online state initialization is very similar to the handshake routine with the following exceptions:

- The *first* interest int:1 issued by the consumer is appended with the encrypted state information (minus the session index) similar to as is done in the handshake variant. The session index SID is still appended to the interest before the state information. This enables the anonymizing proxies to check for the presence of the session index in their state table. If an entry is not found, the remainder of the interest is decrypted and treated as the initial state information.
- All *subsequent* interests int:i $(i > 1)$ are prepended with the session index and *padded* with an l bits of data so that $|\mathsf{int}{:}1| = |\mathsf{int}{:}i|$ (i.e., the length of all interests are equal). Since the concatenation of the session identifier and the state information is indistinguishable from the concatenation of a different session identifer and random pad, an adversary cannot distinguish between int:1 and int:i.

Algorithm 4 AR Content Handling

Require: Content $\overline{data_{i+1}^j}$ in response to interest $\overline{\mathsf{int}}_{i+1}^j$
Ensure: Encrypted data packet $data_i^j$

Recover tuple $T_i = (\overline{\mathsf{int}}_i^j, \overline{\mathsf{int}}_{i+1}^j, \mathsf{session}_i)$ based on $data_{i+1}^j$
Parse $\overline{data_{i+1}^j}$ as a tuple $(data_{i+1}^j, \sigma_{i+1})$
if $\sigma_{i+1} = \epsilon$ and $M_{k_{i+1}} = \epsilon$ **then**
  Pass
else if $\sigma_{i+1} \neq \epsilon$ and $M_{k_{i+1}} \neq \epsilon$ **then**
  if $\sigma_{i+1} = \mathsf{Verify}_{M_{k_{i+1}}}(data_{i+1}^j)$ **then**
    Pass
  else
    return Error
else
  return Error
Remove signature and name from $data_{i+1}^j$
Create new empty data packet $data_i^j$
Set name on $data_i^j$ as the name on $\overline{\mathsf{int}}_i^j$
$data_i^j := \mathsf{Encrypt}_{E_{k_i}}(data_i^j)$
$\sigma_i := \mathsf{MAC}(data_i^j)$
$\overline{data_i^j} = \overline{(data_i^j, \sigma_i)}$
return $\overline{data_i^j}$

Beyond these changes, the operation and usage of the circuits remains the same for consumers and anonymizing proxies. We omit algorithmic details about the above differences since they should be clear from context.

3.4 Interest Flow Control via Windowing

To further increase the throughput of AND, the design makes use of a flow control mechanism analogous to the sliding window technique used in standard TCP implementations. In particular, AND uses a global maximum window size w that refers to the maximum number of interests that can be issued asynchronously without being acknowledged. The behavior of the interest sliding window will then proceed as follows:

- All parties, including the consumer and each AR participating in an anonymous circuit, will maintain four pointers, WindowStart, WindowEnd, KeyStart, and KeyEnd,

Algorithm 1 Circuit and Session Establishment Protocol

Require: Anonymous routers $r_1, r_2, \ldots, r_n$ $(n \geq 1)$ with public keys $pk_1, pk_2, \ldots, pk_n$.

function ServerEstablishSession(int)
  $(k, E_{k_i}, M_{k_i}, M_{k_{i+1}}, \mathsf{EIV}_i, \mathsf{SIV}_i, \mathsf{session}_i, \mathsf{SIndex}_i) := \mathcal{D}_{sk_i}(\mathsf{int})$
  Persist $(\mathsf{session}_i, E_{k_i}, M_{k_i}, M_{k_{i+1}}, \mathsf{EIV}_i, \mathsf{SIV}_i)$ to state, and store $(\mathsf{SIndex}_i, \mathsf{session}_i, \mathsf{SIV}_i)$ in the session table ST_i
  $\mathsf{resp} \leftarrow \mathsf{Encrypt}_k(\mathsf{session}_i, E_{k_i}, M_{k_i}, \mathsf{EIV}_i, \mathsf{SIV}_i)$
  return resp

function ClientEstablishSession(r_i, $M_{k_{i+1}}$)
  $E_{k_i} \leftarrow \{0,1\}^\kappa$
  $M_{k_i} \leftarrow \{0,1\}^\kappa$
  $\mathsf{EIV}_i \leftarrow \{0,1\}^\kappa$
  $x \leftarrow \{0,1\}^\kappa$
  $\mathsf{SIV}_i \leftarrow \{0,1\}^\kappa$
  $\mathsf{session}_i := H(x)$
  $\mathsf{SIndex}_i := H(\mathsf{session}_i + \mathsf{SIV}_i)$
  $\mathsf{int} := \mathsf{namespace}_i/\mathsf{CREATESESSION}/\mathcal{E}_{pk_i}(k, E_{k_i}, M_{k_i}, M_{k_{i+1}}, \mathsf{EIV}_i, \mathsf{SIV}_i, \mathsf{session}_i, \mathsf{SIndex}_i)$
  $\mathsf{resp} := \mathsf{ccnget}(\mathsf{int})$ // reach out to the AR
  return $(E_{k_i}, M_{k_i}, x_i, \mathsf{session}_i, \mathsf{IV}_i)$

function EstablishCircuit($r_1, \ldots, r_n$)
  $(E_{k_n}, M_{k_n}, c_n, \mathsf{session}_n, \mathsf{IV}_n) := \mathsf{ClientEstablishSession}(r_n)$
  for $i = n-1$ **downto** 1 **do**
    if $i = n-1$ **then**
      $(E_{k_i}, M_{k_i}, \mathsf{EIV}_i, \mathsf{session}_i, \mathsf{SIV}_i) := \mathsf{ClientEstablishSession}(r_i, \perp)$
    else
      $(E_{k_i}, M_{k_i}, \mathsf{EIV}_i, \mathsf{session}_i, \mathsf{SIV}_i) := \mathsf{ClientEstablishSession}(r_i, M_{k_{i+1}})$

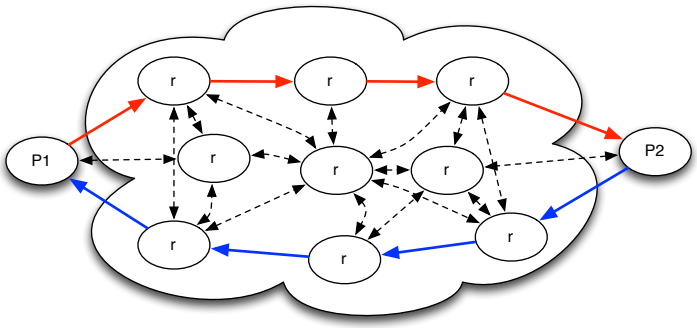

Figure 2: A sample bidirectional circuit configuration in which two parties communicate using mutually exclusive ARs in both directions. Note that it is not required for each circuit to be the same length, nor is it required that the intersection of the routers for each direction of the circuit to be empty (i.e., routers may *unknowingly* support sessions traversing in both directions).

where $\mathsf{WindowEnd} - \mathsf{WindowStart} \leq w$ will always be an invariant. To start, $\mathsf{WindowStart} = \mathsf{WindowEnd} = 0 = \mathsf{KeyStart} = \mathsf{KeyEnd} = 0$.

- When an interest is to be issued or forwarded, the party will check to ensure that $\mathsf{WindowEnd} - \mathsf{WindowStart} < w$, and if so interest is sent/forwarded and WindowEnd is incremented by one. In addition, the keystream at the consumer (for each AR) or AR is pumped by M bits and KeyEnd is incremented by M, where M is the largest piece of content received thus far. Otherwise, the interest is buffered at the consumer or AR.
- When a piece of content arrives that corresponds to some interest issued between WindowStart and WindowEnd, WindowStart is incremented by one and an event is triggered forcing the party to check their interest buffers for new intersts to be sent. In addition, KeyStart is incremented by the size of the content, and checks to see if the maximum content size M needs to be updated.

Using this technique, if a router receives an interest for a particular anonymous circuit when $\mathsf{WindowEnd} - \mathsf{WindowStart} \geq w$, then the interest is dropped. Interest and content flow control and correct functionality of the consumer guarantees that interests will not be issued when the window is full.

Algorithm 3 AR Encrypted Interest Forwarding

Require: $\overline{\mathsf{int}}_i^j$
Ensure: $(\overline{\mathsf{int}}_{i+1}^j, \mathsf{session}_i)$ or discarded packet
if $\mathsf{SIndex}_i \in \mathsf{ST}_i$ **then**
  Let $(\mathsf{session}_i, E_{k_i}, M_{k_i}, \mathsf{EIV}_i, \mathsf{SIV}_i)$ be the session information associated with SIndex_i
  $\mathsf{SIV}_i := \mathsf{SIV}_i + 1 \pmod{2^\kappa}$
  $\mathsf{SIndex}_i := H(\mathsf{session}_i + \mathsf{SIV}_i)$
  Update $(\mathsf{SIndex}_i, \mathsf{session}_i, \mathsf{SIV}_i)$ in the session table ST_i
  $(\overline{\mathsf{int}}_{i+1}^j, timestamp) := \mathsf{Decrypt}_{E_{k_i}}(\overline{\mathsf{int}}_i^j)$
  if decryption fails or $\mathsf{timestamp}$ is not current **then**
    Discard $\overline{\mathsf{int}}_i^j$
  else
    Persist tuple $T_i = (\overline{\mathsf{int}}_i^j, \overline{\mathsf{int}}_{i+1}^j, \mathsf{session}_i)$ to pending interest table PT_i
    return $(\overline{\mathsf{int}}_{i+1}^j, \mathsf{session}_i)$
else
  Discard $\overline{\mathsf{int}}_i^j$

Also, while all parties must share the same window size, the consumer w is free to change the window size to adapt to the state of the network. For instance, if there is very little end-to-end latency for content, then the consumer may wish to increase the size of the window. In order to support such dynamic windowing, the consumer must inform all routers in an anonymous circuit of the desired window size. AND supports this by appending the window size to the session index SIndex_i for each r_i in a circuit. In order to ensure that no two interests are distinguishable by this window size w, it is also masked by a random one-time pad p (i.e. $w' = w \oplus p$, where w is interpreted as an integer encoded in binary) that is included in the encrypted interest. In other words, once an interest is decrypted the pad p is recovered, and then the router computes $w = w' \oplus p$. If w is larger than the stored value associated with the circuit, then the new result is simply updated and operation proceeds as normal. Otherwise, the router enters a "backoff" state where it waits for content to be returned and the resulting window size to return to normal before issuing any further interests. This dynamic windowing method is currently not implemented in the AND system.

4. IMPLEMENTATION

AND is implemented entirely in C using the CCNx 0.82 library [6]. Anonymizing routers run instances of the `AnonServer` application with a single prefix URI, whereas consumers run instances of the `AnonConsumer` application with a set of configuration options and sequence of prefix URIs denoting the circuit to be constructed. The `AnonServer` module instantiates a single `DownstreamProxy`, which is a subclass of the base `Proxy` class used to interface with the CCN daemon running on the host. This downstream proxy is responsible for unwrapping incoming interests and wrapping upstream content. It is also responsible for maintaining the session state information associated with each circuit, which includes populating the state table for both the handshake and online session establsihment variants.

The `AnonConsumer` application creates a circuit of `UpstreamProxy` instances, which are also subclasses of the base `Proxy` class, that are responsible downstream content encryption, as well as a single `UpstreamProxy` that is responsible for generating the wrapped version of each incoming client interest. Upon instantiation, each `UpstreamProxy` establishes session state and either persists it for the online session establishment or shares it with the corresponding `DownstreamProxy` instance running on the anonymizing proxy if the handshake variant is used.

The majority of the functionality is encapsulated in the four procedures, `WrapInterest`, `UnwrapInterest`, `WrapContent`, and `UnwrapContent`. Additional functionality, such as interest window maintenance (see Section 3.4), is implemented by the respective proxy classes in conjunction with helper classes such as `ProxyState`.

5. SECURITY ANALYSIS

In order to assess the security of AND it is important to first define an adversarial model and corresponding definition of security. To this end, we define an adversarial model for AND that has the same capabilities as presented in [2]: Deploy compromised routers, Compromise existing routers, Control content producers, Deploy compromised caches, and Observe and replay traffic. Furthermore, any of these actions or capabiltiies can be carried out adaptively (i.e., in response to status updates from the network or based on the adversary's observations). We also note that the time required to carry out an attack is non-negligibly larger than the average RTT for an interest-content exchange in order to make this model realistic.

We also emphasize that the fundamental differences between the design of ANDāNA and AND are that, in AND, each adjacent router will share a private MAC key used for efficient content signature generation (and, optionally, verification) and sessions are identified by the output of H (rather than encrypting and decrypting interests using expensive asymmetric procedures). Accordingly, the proofs of anonymity and privacy need to be augmented to take this into account. The remainder of the design is syntactically equivalent to that of ANDāNA, and so we may restate the theoreoms without proof. However, in doing so, we generalize them to circuits of length $n \geq 2$.

Theorem 1. *Consumer* $u \in (\mathsf{C} \backslash \mathsf{C}_\mathcal{A})$ *has consumer anonymity*

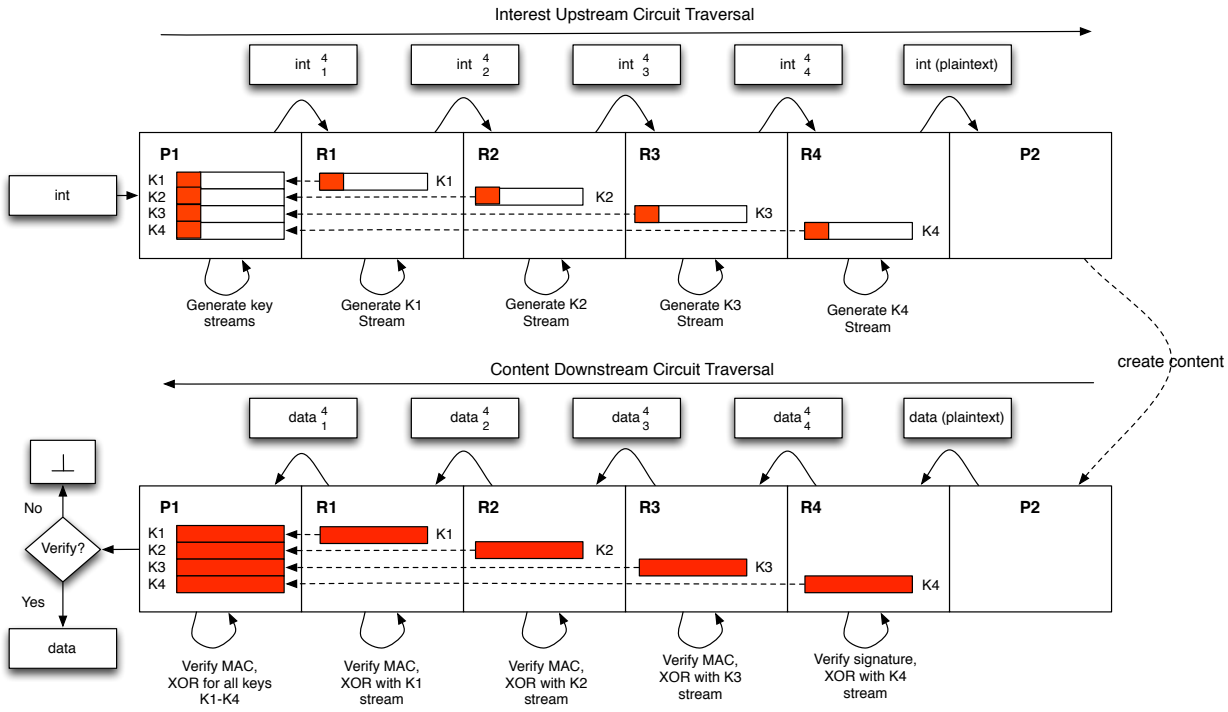

Figure 3: Visual depiction of window-based keystream precomputation during upstream interest traversal.

in configuration C with respect to adversary $\mathcal{A}$ if there exists $u \neq u'$ such that any of the following conditions hold:

1. $u, u' \in \mathsf{AS}_{\mathcal{A}}^{C_4(u)}$
2. *There exists ARs r_i and r_i' such that $r_i, r_i' \notin \mathsf{R}_{\mathcal{A}}$, both r_i and r_i' are on the circuit traversed by $C_4(u) = \overline{\mathsf{int}}_1^n$.*

Proof. See [2]. □

Theorem 2. *Consumer u has producer anonymity in configuration C with respect to producer $p \in \mathsf{P}$ and adversary $\mathcal{A}$ if there exists a pair of ARs r_i and r_i' such that r_i and r_i' (for some uncompromised entity $u \notin \mathsf{C}_{\mathcal{A}}$) are on the path traversed by $C_4(u) = \overline{\mathsf{int}}_1^n$, $C_1(u) = C_1(u')$, and $C_3(u) = p \neq C_3(u')$.*

Proof. See [2]. □

Corollary 1. *Consumer $u \in (\mathsf{C} \backslash \mathsf{C}_{\mathcal{A}})$ and producer $p \in \mathsf{P}$ are unlinkable in configuration C with respect to adversary $\mathcal{A}$ if p has producer anonymity with respect to u's interests or u has consumer anonymity and there exists a configuration $C' \equiv_{\mathcal{A}} C$ where $C'(u') = C(u)$ with $u' \neq u$ and u''s interests have a destination different from p.*

In addition to security, we must also be concerned about the correct functioning of each AR supporting a session between two parties. In this context, we (informally) define session correctness as the ability of a consumer to correctly decrypt content that is generated *in response to* its original interest. That is, if a consumer issues an interest, it should be able to correctly decrypt the content that it receives. The following factors impact the correctness of the session:

1. Each AR $r_1, \ldots, r_n$ on the consumer-to-producer circuit should correctly recover the session identifier associated with the current session.
2. The session key streams should only be advanced upon the receipt of an interest corresponding to the consumer who initiated the session or content that is generated from the upstream router (potentially the producer) in the circuit.

The first item is necessary in order for each AR to correctly decrypt interests, encrypt content, and perform content signature generation and verification. The second item is necessary so that all content can be correctly decrypted by the consumer. We claim that, given a CCA-secure public key encryption scheme, the probability that either one of these factors being violated by an adversary $\mathcal{A}$ is negligible. Let ForgeSession and KeyJump denote the events corresponding to instances where an adversary creates a ciphertext that maps to a valid session identifier for *some* session currently supported by an AR (i.e., the forged session belongs to the routers session table ST), and the event that an adversary causes the key stream for *some* AR in a consumer-to-producer circuit to fall out of sync with the consumer. By the design of AND, it should be clear that KeyJump occurs when ForgeSession occurs, since the key

stream is only advanced upon receipt of an interest, but may also occur when an adversary successfully forges a MAC tag corresponding to the signature of a piece of content from the upstream router (or producer). We denote this latter event as ContentMacForge. With the motivation in place, we now formally analyze the probabilities of these events occuring below. For notational convenience, we assume that each event only occurs as a result of some adversarial action, so we omit this relation in what follows.

Lemma 1. *For all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists some negligible function* negl *such that*

$$\Pr[\mathsf{ForgeSession}] \leq \mathsf{negl}(\kappa).$$

Proof. By the design of AND, we know that session identifiers are computed as the output of a collision resistant hash function $H : \{0,1\}^* \to \{0,1\}^m$, where $m = \mathsf{poly}(\kappa)$ (i.e. polynomial in the global security parameter). Consequently, forging a session identifier *without* the input to H implies that a collision was found, thus violating collision resistance of H. Thus, forging a session is equally hard as finding a collision in H, or more formally, $\Pr[\mathsf{Collision}(H) = 1] = \Pr[\mathsf{ForgeSession}]$. By the properties of collision resistance of H which states that $\Pr[\mathsf{Collision}(H) = 1] \leq \mathsf{negl}(\kappa)$ for some negligible function negl, it follows that $\Pr[\mathsf{ForgeSession}] \leq \mathsf{negl}(\kappa)$.

□

Lemma 2. *For all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists some negligible function* negl *such that*

$$\Pr[\mathsf{ContentMacForge}] \leq \mathsf{negl}(\kappa).$$

Proof. By the design of AND, the MAC scheme Π used for content symmetric content signature generation and verification is defined as $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Ver})$, where Gen generates the secret key k used in the scheme, $\mathsf{Mac}_k(m)$ outputs the MAC tag $t := F_k(m)$ for some pseudorandom function F, and $\mathsf{Ver}_k(m,t)$ outputs 1 if $t = \mathsf{Mac}_k(m)$ and 0 otherwise. This is known and proven to be a secure MAC scheme [does this warrant citation?], meaning that for all probabilistic polynomial-time adversaries $\mathcal{A}$ there exists a negligible function negl such that $\Pr[\mathsf{MacForce}_{\mathcal{A},\Pi}(1^\kappa) = 1] \leq \mathsf{negl}(\kappa)$, and since ContentMacForce occurs exactly when the even MacForce occurs, we have that $\Pr[\mathsf{ContentMacForge}] \leq \mathsf{negl}(\kappa)$.

□

Lemma 3. *For all probabilistic polynomial-time adversaries $\mathcal{A}$, there exists some negligible function* negl *such that*

$$\Pr[\mathsf{KeyJump}] \leq \mathsf{negl}(\kappa).$$

Proof. By the design of AND, it follows that $\Pr[\mathsf{KeyJump}] = \Pr[\mathsf{ForgeSession}] + \Pr[\mathsf{ContentMacForce}]$, and since the sum of two negligible functions is also negligible, it follows that there exists some negligible function negl such that $\Pr[\mathsf{KeyJump}] \leq \mathsf{negl}(\kappa)$. □

Theorem 3. *Session correctness of* AND *is only violated with negligible probability.*

Proof. This follows immediately from Lemmas 1, 2, and 3 and the fact that the sum of two negligible functions is also negligible.[1] □

6. PERFORMANCE

TODO

7. REFERENCES

[1] V. Adhikari, Y. Guo, F. Hao, M. Varvello, V. Hilt, M. Steiner, and Z.-L. Zhang. Unreeling netflix: Understanding and improving multi-cdn movie delivery. In *INFOCOM, 2012 Proceedings IEEE*, pages 1620–1628, March 2012.

[2] S. DiBenedetto, P. Gasti, G. Tsudik, and E. Uzun. Andana: Anonymous named data networking application. In *Network and Distributed System Security - NDSS*. The Internet Society, 2012.

[3] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[4] M. Franz, B. Meyer, and A. Pashalidis. Attacking unlinkability: The importance of context. In N. Borisov and P. Golle, editors, *Privacy Enhancing Technologies*, volume 4776 of *Lecture Notes in Computer Science*, pages 1–16. Springer Berlin Heidelberg, 2007.

[5] S. Murdoch and G. Danezis. Low-cost traffic analysis of tor. In *Security and Privacy, 2005 IEEE Symposium on*, pages 183–195, May 2005.

[6] PARC. CCNx. Available online at: `https://github.com/ProjectCCNx/ccnx`, May 2014.

[7] S. Schiffner and S. Clauß. Using linkability information to attack mix-based anonymity services. In *Proceedings of the 9th International Symposium on Privacy Enhancing Technologies*, PETS '09, pages 94–107, Berlin, Heidelberg, 2009. Springer-Verlag.

[8] L. Zhang, D. Estrin, J. Burke, V. Jacobson, J. D. Thornton, D. K. Smetters, B. Zhang, G. Tsudik, D. Massey, C. Papadopoulos, et al. Named data networking (ndn) project. *Relatório Técnico NDN-0001, Xerox Palo Alto Research Center-PARC*, 2010.

[1] This sum comes from the fact that the probability of the "failure" events occurring must be taken into account in both directions of the session.