ANDāNA-v2: Application-Layer Support for Low-Latency Bidirectional Anonymous Traffic in NDN

Christopher A. Wood*
Institute for Clarity in Documentation
1932 Wallamaloo Lane
Wallamaloo, New Zealand
trovato@corporation.com

Gene Tsudik†
Institute for Clarity in Documentation
P.O. Box 1212
Dublin, Ohio 43017-6221
webmaster@marysville-ohio.com

Ersin Uzun‡
The Thørväld Group
1 Thørväld Circle
Hekla, Iceland
larst@affiliation.org

ABSTRACT
TODO

Categories and Subject Descriptors
H.4 [**Information Systems Applications**]: Miscellaneous; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms
TODO

Keywords
TODO

1. INTRODUCTION
Usage of the Internet has undergone a tremendous trasnformation 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 [**?**]. 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) [**?**] 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 has strong implications on content security and consumer privacy.

TODO: introduce privacy notions from ANDANA paper, introduce ANDANA application (below), and then motivate ANDANAv2

The primary motivation for a new desgin 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 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 [**?**]. 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 anony-

*Dr. Trovato insisted his name be first.
†The secretary disavows any knowledge of this author's actions.
‡This author is the one who did all the really hard work.

mous 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 [**?**], 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 [**?**].

As most of the literature focuses on mix-based anonymizing services like 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, the formal goal for ANDāNA-v2 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 high-throughput and low-latency traffic between two parties.

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-v1), 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.

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
(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 ($i \leq$
$\mathcal{A}$	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 tuple
H	A collision resistant hash function on the domain $\{0,1\}^*$
F_k	A keyed pseudorandom function

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 discusse in the following section, this lack of information is insufficient with regards to consumer privacy.

2.2 ANDāNA-v1 Design Highlights and Pitfalls

ANDāNA was originally designed as a proof-of-concept equivalent of Tor [**?**] in the context of NDN.

3. ANDANA-V2 DESIGN

In this section we describe the much improved design of ANDāNAv2. Table 1 introduces the relevant notation needed to understand the design in its entirety

3.1 Circuit and Session Establishment

At the heart of the ANDāNAv2 is the notion of anonymizing routers and connection-oriented circuits, similar in spirit to the inner workings of TOR [**?**]. Anonymizing routers serve two purposes in ANDāNAv2: (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 ANDāNAv2 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 ANDāNAv2 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 ANDāNAv2, we see that the following factors weigh in on the overall performance of the application:

1. Content encryption
2. Content signature generation
3. Encrypted interest generation

In ANDāNA-v2 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} used to sign (and optionally 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 1. 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.

Notice that by this procedure, no two routers will share the same session identifier even though they partake in the same circuit since they generate session identifiers independent and uniformly at random from $\{0,1\}^\kappa$. Also, the ServerEstablishSession and ServerRetrieveMACKey procedures are asynchronous and only invoked in response to the respective interest.

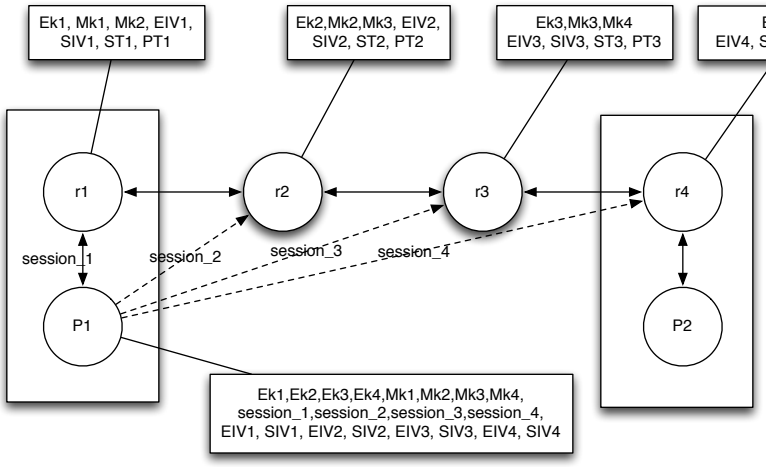

Figure 1: Sample session-based circuit between a consumer C and producer P with ARs r_1, r_2, r_3, r_4, where the end routers on the path are run on the same node as C and P.

3.2 ANDāNA-v2 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 ANDāNAv2 design. Specifically, there are three main operations that need to be defined: encrypted interest generation,

Algorithm 1 Circuit and Session Establishment Protocol

Require: Anonymous routers $r_1, r_2, \ldots, r_n$ $(n \geq 2)$ with public keys $pk_1, pk_2, \ldots, pk_n$.

function ServerRetrieveMACKey(int) // Server-side at router i
  $(M_k, \mathsf{session}_i, x) := \mathcal{D}_{sk_i}(\mathsf{int}[-1])$ // Recover MAC key and randomness x
  if $\mathsf{session}_i$ not in state **then**
    return Error
  else
    Persist M_k as $M_{k_{i+1}}$ (the upstream MAC key) with session $\mathsf{session}_i$
    $x^* := \mathsf{MAC}_{M_k}(x)$
    $\mathsf{resp} := x^*$
    return resp
function ServerEstablishSession(int) // Server-side at router i
  $k \leftarrow \mathcal{D}_{sk_i}(\mathsf{int}[-1])$ // Recover session encryption key
  $E_{k_i} \leftarrow \{0,1\}^\kappa$ // Encryption key
  $M_{k_i} \leftarrow \{0,1\}^\kappa$ // MAC key
  $\mathsf{EIV}_i \leftarrow \{0,1\}^\kappa$ // counter IV
  $x \leftarrow \{0,1\}^\kappa$
  $\mathsf{SIV}_i \leftarrow \{0,1\}^\kappa$ // session IV
  $\mathsf{session}_i := H(x)$ // session ID
  $\mathsf{SIndex}_i := H(\mathsf{session}_i + \mathsf{SIV}_i)$
  Persist $(\mathsf{session}_i, E_{k_i}, M_{k_i}, \mathsf{EIV}_i, \mathsf{SIV}_i)$ 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 ClientSendMACKey(r_i, $\mathsf{session}_i$, M_k) // Client-side
  $x \leftarrow \{0,1\}^\kappa$
  $x' := \mathsf{MAC}_{M_k}(x)$
  $T \leftarrow \mathcal{E}_{pk_i}(M_k, \mathsf{session}_i, x)$
  $\mathsf{int} := \mathsf{namespace}_i/\mathsf{SESSIONMAC}/T$
  $\mathsf{resp} := \mathsf{ccnget}(\mathsf{int})$ // reach out to the AR
  $x^* := \mathsf{resp}[-1]$
  if $x' \neq x^*$ **then**
    return Fail
  else
    return Pass
function ClientEstablishSession(r_i) // Client-side
  $k \leftarrow \{0,1\}^\kappa$
  $\overline{k} \leftarrow \mathcal{E}_{pk_i}(k)$
  $\mathsf{int} := \mathsf{namespace}_i/\mathsf{CREATESESSION}/\overline{k}$
  $\mathsf{resp} := \mathsf{ccnget}(\mathsf{int})$ // reach out to the AR
  $(E_{k_n}, M_{k_n}, \mathsf{EIV}_n, \mathsf{session}_n, \mathsf{SIV}_n) \leftarrow \mathsf{Decrypt}_k(\mathsf{resp})$
  return $(E_{k_n}, M_{k_n}, c_n, \mathsf{session}_n, \mathsf{IV}_n)$
function EstablishCircuit($r_1, \ldots, r_n$) // Main procedure
  $(E_{k_n}, M_{k_n}, c_n, \mathsf{session}_n, \mathsf{IV}_n) :=$ ClientEstablishSession(r_n)
  for $i = n-1$ **downto** 1 **do**
    $(E_{k_i}, M_{k_i}, \mathsf{EIV}_i, \mathsf{session}_i, \mathsf{SIV}_i) :=$ ClientEstablishSession(r_n)
    if ClientSendMACKey($r_i, \mathsf{session}_i, M_{k_{i+1}}$) = Fail **then**
      return Fail

AR interest forwarding, and AR content handling. In what we follows we present the details of each of these procedures as needed for ANDāNA-v2. 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 $\overline{\mathsf{int}}_1^n$.

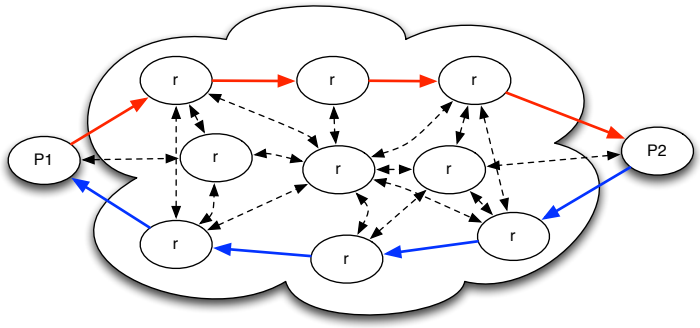

Figure 2: A sample bidirectional circuit configuration in which two parties communicate using mutually exclusive ARs in both directions. Note that it is not required for each circuit to be the same length, nor is it required that the intersection of the routers for each direction of the circuit to be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

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 $\overline{\mathsf{int}}_1^n$

4. IMPLEMENTATION

TODO

5. SECURITY ANALYSIS

In order to assess the security of ANDāNA-v2 it is important to first define an adversarial model and corresponding definition of security. To this end, we define an adversarial model for ANDāNA-v2 that has the same capabilities as presented in [**?**]:

- Deploy compromised routers
- Compromise existing routers
- Control content producers
- Deploy compromised caches

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
1: **if** $\mathsf{SIndex}_i \in \mathsf{ST}_i$ **then**
2: Let $(\mathsf{session}_i, E_{k_i}, M_{k_i}, \mathsf{EIV}_i, \mathsf{SIV}_i)$ be the session information associated with SIndex_i
3: $\mathsf{SIV}_i := \mathsf{SIV}_i + 1 \pmod{2^\kappa}$
4: $\mathsf{SIndex}_i := H(\mathsf{session}_i + \mathsf{SIV}_i)$
5: Update $(\mathsf{SIndex}_i, \mathsf{session}_i, \mathsf{SIV}_i)$ in the session table ST_i
6: $(\overline{\mathsf{int}}_{i+1}^j, timestamp) := \mathsf{Decrypt}_{E_{k_i}}(\overline{\mathsf{int}}_i^j)$
7: **if** decryption fails or timestamp is not current **then**
8: Discard $\overline{\mathsf{int}}_i^j$
9: **else**
10: Persist tuple $T_i = (\overline{\mathsf{int}}_i^j, \overline{\mathsf{int}}_{i+1}^j, \mathsf{session}_i)$ to pending interest table PT_i
11: **return** $(\overline{\mathsf{int}}_{i+1}^j, \mathsf{session}_i)$
12: **else**
13: Discard $\overline{\mathsf{int}}_i^j$ ▷ Some form of interest flooding prevention should be employed here

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 $data_i^j$
1: Recover tuple $T_i = (\overline{\mathsf{int}}_i^j, \overline{\mathsf{int}}_{i+1}^j, \mathsf{session}_i)$ based on $data_{i+1}^j$
2: Parse $\overline{data_{i+1}^j}$ as a tuple $(data_{i+1}^j, \sigma_{i+1})$
3: **if** $\sigma_{i+1} = \epsilon$ and $M_{k_{i+1}} = \epsilon$ **then** ▷ Last hop router - does not verify MAC
4: Pass
5: **else if** $\sigma_{i+1} \neq \epsilon$ and $M_{k_{i+1}} \neq \epsilon$ **then**
6: **if** $\sigma_{i+1} = \mathsf{Verify}_{M_{k_{i+1}}}(data_{i+1}^j)$ **then**
7: Pass
8: **else**
9: **return** Error ▷ MAC verification did not pass
10: **else** ▷ There was either a tag or we don't have the upstream MAC key - either way, we error
11: **return** Error
12: Remove signature and name from $data_{i+1}^j$
13: Create new empty data packet $data_i^j$
14: Set name on $data_i^j$ as the name on $\overline{\mathsf{int}}_i^j$
15: $data_i^j := \mathsf{Encrypt}_{E_{k_i}}(data_i^j)$
16: $\sigma_i := \mathsf{MAC}(data_i^j)$
17: $\overline{data_i^j} = \underline{((data_i^j, \sigma_i))}$
18: **return** $\overline{data_i^j}$

- 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.

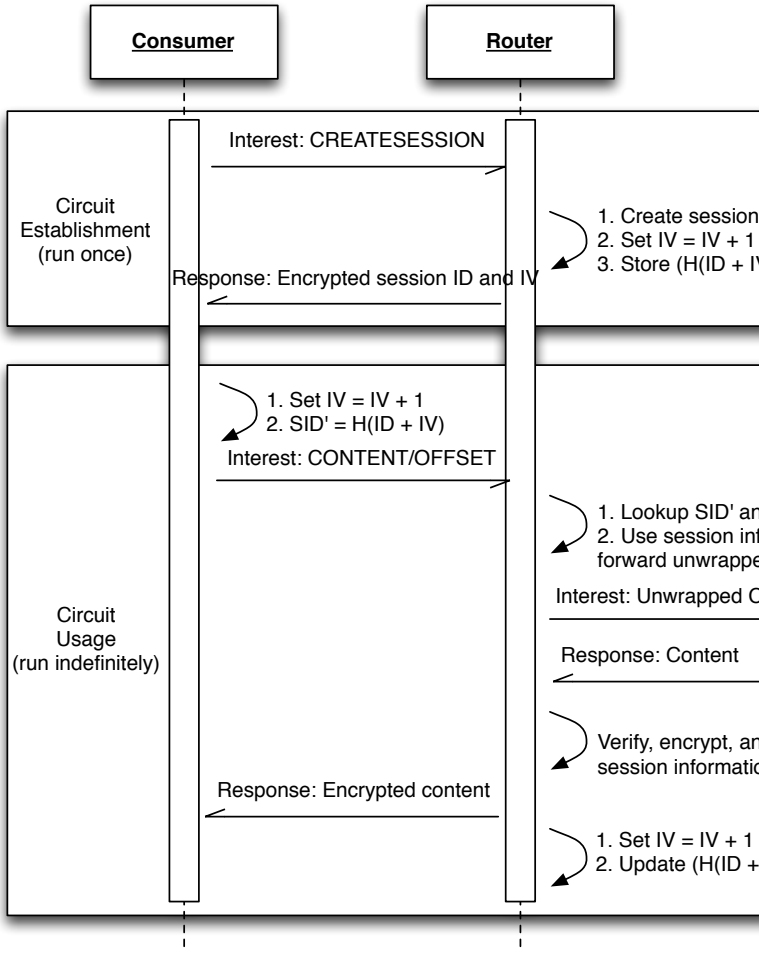

Figure 3: Visual depiction of the interaction between the consumer and the first hop router. The procedure repeats in the same manner for all further upstream routers after each interest is unrolled and resulting content is encrypted.

We also make use of the same notions of producer and consumer anonymity and unlinkability to define the security of this scheme. Before reintroducing these definitions, we first establish the relevant notation. An adversary $\mathcal{A}$ is defined as a 4-tuple $(\mathsf{P}_\mathcal{A}, \mathsf{C}_\mathcal{A}, \mathsf{R}_\mathcal{A}, \mathsf{IF}_\mathcal{A})$, where each individual component denotes the set of producers, consumers, routers, and interfaces compromised by $\mathcal{A}$, respectively. Following [**?**], a router r is deemed compromised (i.e., $r \in \mathcal{R}_\mathcal{A}$) if all of its interfaces belong to $\mathsf{IF}_\mathcal{A}$. Similarly, if $\mathcal{A}$ controls a producer or consumer then they have complete (and adaptive) control over how they behave in the application session, meaning that $\mathcal{A}$ can control everything from the timing, format, and actual information of each piece of content. We also define the anonmity set with respect to an interface if_i^r (i.e., interface i of router r) to be

$$\mathsf{AS}_{\mathsf{if}_i^r} = \{d | \Pr[d \rightarrow_{\mathsf{int}} r | \mathsf{int} \rightarrow \mathsf{if}_i^r] > 0\},$$

and the anonymity set with respect to the adversary $\mathcal{A}$ to be

$$\mathsf{AS}_{\mathcal{A}}^{\mathsf{int}} = \bigcap_{\mathsf{path}^{\mathsf{int}} \cap \mathsf{IF}_{\mathcal{A}}} \mathsf{AS}_{\mathsf{if}_i^r},$$

where $\mathsf{path}^{\mathsf{int}}$ is the set of interfaces along which the interest int traversed in the circuit from the consumer to the producer.

Finally, we denote the "state" of a network, or configuration, as a snapshot in time of its current activity. Accordingly, each configuration is a relation that maps parties to their actions (e.g., interest or content creation). For circuits of length n, let a configuration C be defined as

$$C : \mathsf{C} \rightarrow \{(r_1, \ldots, r_n, p, \overline{\mathsf{int}}_1^n)\},$$

where the $(n+2)$-element tuple $(r_1, \ldots, r_n, p, \mathsf{int}_1^n) \in \mathsf{R}^n \times \mathsf{P} \times \{0,1\}^*$. This relation can be viewed as a map from a consumer $c \in \mathsf{C}$ to a set of routers defining the circuit from c to all producers $p \in \mathsf{P}$ along which interests $\overline{\mathsf{int}}_1^n$ traverse.

Following in the footsteps of [**?**], we define the security of our design in the context of indistinguishable network configurations. Specifically, two configurations C and C' are said to be *indistinguishable with respect to* $\mathcal{A}$, denoted $C \equiv_{\mathcal{A}} C'$, if for all such polynomial-time adversaries $\mathcal{A}$ there exists a negligible function ϵ such that

$$\left|\Pr[\mathcal{A}(1^n, C) = 1] - \Pr[\mathcal{A}(1^n, C') = 1]\right| \leq \epsilon(\kappa),$$

for the global security parameter κ.

We now define security of our design in terms of consumer and producer anonymity and unlinkability. These can be found in [**?**], but we provide them here for completeness. We also provide new definitions that make sense in the context of the bidirectional session case that ANDāNA-v2 is intended to support.

Definition 1. *[?] 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. *[?] 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. *[?] 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 exists a configuration $C' \equiv_{\mathcal{A}} C$ where u's interests are sent to a producer $p' \neq p$.*

We emphasize that the fundamental differences between the design of ANDāNA and ANDāNA-v2 are that, in ANDāNA-v2, 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 $u \in (\mathsf{C} \setminus \mathsf{C}_{\mathcal{A}})$ has consumer anonymity 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 [**?**]. □

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 [**?**]. □

Corollary 1. *Consumer $u \in (\mathsf{C} \setminus \mathsf{C}_{\mathcal{A}})$ 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 ANDāNA-v2, 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 ANDāNA-v2, 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 ANDāNA-v2, 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 $\Pr[\mathsf{MacForge}_{\mathcal{A},\Pi}(1^\kappa) = 1] \leq \mathsf{negl}(\kappa)$, 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 ANDāNA-v2, 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* ANDāNA-v2 *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. CONCLUSION

TODO

8. ACKNOWLEDGMENTS

TODO

[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.