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

Christopher A. Wood
UC Irvine
woodc1@uci.edu

November 24, 2013

Abstract

This report documents the motivation, design, and implementation plan for the new version of ANDāNA to support highly efficient bidirectional traffic with low-latency in NDN. Weeks 1-3 of this project were spent setting up the experimental testbed and ramping up with the ANDāNA design and source code; weeks 4-6 were spent designing and running experiments to gather performance data for ANDāNA; weeks 7-8 were spent on the design of ANDāNA-v2 and preliminary attempts at its implementation; weeks 9-10 were spent finalizing the design, completing the implementation, and gathering experimental data that can be used to compare against ANDāNA.

1 Overview

With the possibile adoption of NDN or any flavor of information-centric networking as the future Internet architecture in the near future, support for low-latency, bidirectional traffic may prove very useful for a variety of use cases. Consider the scenario in which two parties wish to exchange voice or video content over NDN, such as the case with Skype and related applications. Jacobson et al. [1] have already studied the efficiencies of such content-heavy and QoS-strict applications over CCN. Unfortunately, support for such applications becomes much more difficult when one, or both, of the parties wishes to remain anonymous. Such is the case in settings where voice or video content is being streamed from *some* person in a single organization, but the identity of said person needs to remain secret. A real-world instance where this may be useful is when such traffic needs to be exchanged between military organizations of unfriendly nations. Indeed, the identity of military personell participating in such voice or video conferences should remain secret for safety reasons.

Application-layer support for anonymizing network traffic has already been implemented in the ANDāNA system [2]. However, in that work, the authors focus on a single, unidirectional client-producer scenario in which traffic latency, network jitter, and quality-of-service requirements were not specified nor mandated. In this work, we seek to amend the design of ANDāNA to support low-latency, bidirectional traffic applications with hard QoS and anonymity requirements. Consequently, this new design seeks to attain equivalent anonymity and privacy guarantees of ANDāNA while enjoying significantly higher performance - two factors that are often inversely related in practice.

In the remainder of this document we discuss the motivation and details of our preliminary design for the new anonymizing application, dubbed ANDāNA-v2, as well as our implementation strategy. In order to avoid confusion and repetition throughout the rest of the document, all common terminology used throughout is specified and clarified in Table 1. Also, for simplicity, we denote the global system security parameter as κ, unless otherwise specified.

2 Motivation

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,

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 $(i \leq j)$
$\mathcal{A}$	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

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 [9]. 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 [7], 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 [8].

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.

3 Circuit and Session Establishment

At the heart of the ANDāNA 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 ANDāNA: (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 ANDāNA 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 ANDāNA 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 ANDāNA, 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 4. 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.

3.1 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

Algorithm 1 Circuit and Session Establishment Protocol

Input: 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
  $(E_{k_n}, M_{k_n}, c_n, \mathsf{session}_n, \mathsf{IV}_n) := \mathsf{ClientEstablishSession}(r_n)$
  for $i = n-1$ **downto** 1 **do**
    $(E_{k_i}, M_{k_i}, \mathsf{EIV}_i, \mathsf{session}_i, \mathsf{SIV}_i) := \mathsf{ClientEstablishSession}(r_n)$
    if $\mathsf{ClientSendMACKey}(r_i, \mathsf{session}_i, M_{k_{i+1}}) = \mathsf{Fail}$ **then**
      return Fail

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 ANDāNA design. Specifically, there are three main operations that need to be defined: encrypted interest generation, AR

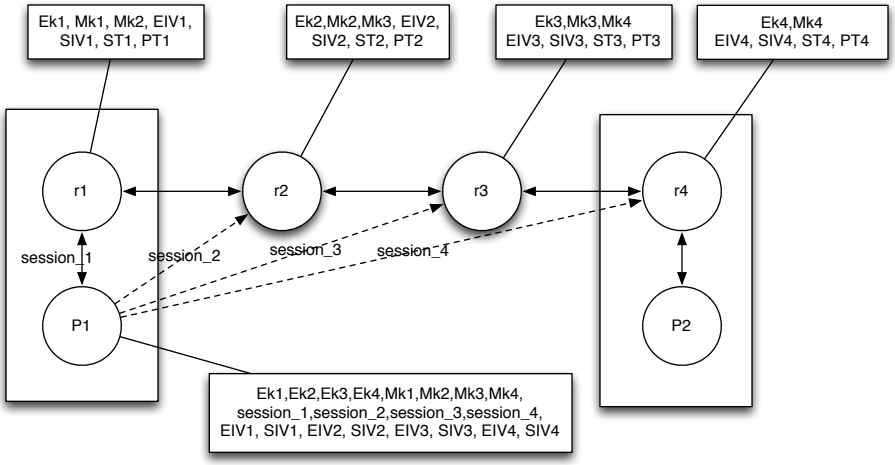

Figure 1: Sample session-based circuit between a consumer C and producer P with ARs r_1, r_2, r_3, r_4, where the end routers on the path are run on the same node as C and P.

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

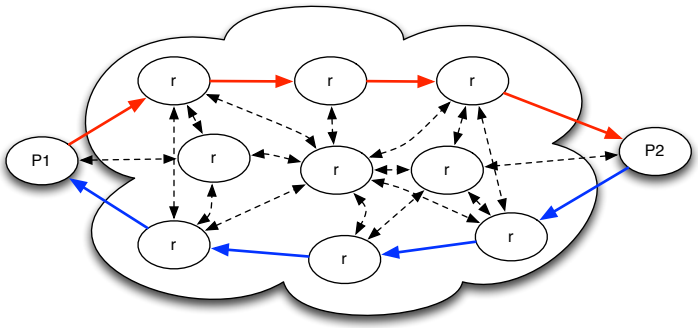

Figure 2: A sample bidirectional circuit configuration in which two parties communicate using mutually exclusive ARs in both directions. Note that it is not required for each circuit to be the same length, nor is it required that the intersection of the routers for each direction of the circuit to be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

4 Instantiating Cryptographic Primitives

Given the generic design presented in the above section, we now need to consider exactly which cryptographic primitives will be used to instantiate this design in usable software. Observe that we require the following

Algorithm 2 Encrypted Interest Generation

Input: Interest int, circuit length n, AR pool $\mathcal{R}$
Output: 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 $\overline{\mathsf{int}}_1^n$

Algorithm 3 AR Encrypted Interest Forwarding

Input: $\overline{\mathsf{int}}_i^j$
Output: $(\overline{\mathsf{int}}_{i+1}^j, \mathsf{session}_i)$ 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 $\overline{\mathsf{int}}_i^j$ ▷ Some form of interest flooding prevention should be employed here

primitives:

1. CCA-secure symmetric key encryption scheme $\Pi_E = (\mathsf{Gen}_E, \mathsf{Encrypt}, \mathsf{Decrypt})$ (content encryption and decryption.
2. Arbitrary input and output collision resistant hash function H (content signing and session identification).
3. CMA-EF secure message authentication code (MAC) scheme $\Pi_M = (\mathsf{Gen}_M, \mathsf{Mac}, \mathsf{Verify})$ and (content signing)

AES-GCM will be used as the CCA-secure symmetric key encryption scheme Π_E, the Keccak hash function will be used for the arbitrary length input and fixed-length (256-bit) output hash function H, and the highly efficient SipHash PRF (as a CMA-EF secure MAC) will serve as the basis for the MAC scheme Π_M.

With these particular cryptographic primitive instantiations, we now analyze the performance bottlenecks of the ANDāNA-v2 design. We do so on a case-by-case basis below, noting optimizations that can be implemented in the design along the way.

Content encryption: AES-GCM operates in counter-mode for CPA-secure encryption and uses the GHASH hash function, based on finite field operations in $GF(2^{128})$, to generate a tag based on the corresponding ciphertext. For efficiency reasons, the "key stream" of the block cipher can be precomputed prior to the arrival of content, or, alternatively, while an interest is being handled upstream (see Figure 4), meaning that the ciphertext and tag generation complexity for a ciphertext of n 128-bit blocks reduces to n XOR operations and n GHASH computations. Clearly, the GHASH computation dominates this

Algorithm 4 AR Content Handling

Input: Content $\overline{data_{i+1}^j}$ in response to interest $\overline{\mathsf{int}}_{i+1}^j$
Output: 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** ▷ Last hop router - does not verify MAC
    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 ▷ MAC verification did not pass
else ▷ There was either a tag or we don't have the upstream MAC key - either way, we error
    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 $\overline{data_i^j}$

complexity, and thus the efficiency of content encryption reduces to highly efficient multiplcation in $GF(2^{128})$.

Session identification: Computing session identifiers (that are sent in the clear) require a single modular addition and hash function invocation. Clearly, the efficiency of the Keccak hash function dominates in this regard.

Interest encryption: Interest encryption will be done using AES-GCM with the same content encryption key. As such, with pre-computed key streams (i.e., encryptions of the counter values) done at each router, the efficiency of this step reduces to the efficiency of the GHASH function. [1]

Content signature generation/verification: The complexity of both of these procedures is clearly dominated by the efficiency of the Keccak hash function. Symmetric MAC tag generation and verification are far more efficient than the Keccak hash function.

We now evaluate these cryptographic algorithms with respect to their performance. Although the performance of all of the proposed cryptographic algorithms are well studied in the literature, we performed our own experiemental analysis that uses the same baseline machine for the benchmarking. Tables 4 and 3 detail these relevant performance metrics (in terms of cycles/byte for each piece of input) to establish a sense of the relative computational complexity of each algorithm.

As expected, the block ciphers and SipHash PRF are the highest performing algorithms used in the design of ANDāNA-v2, especially with platform-specific instructions available for such cryptographic procedures. We note that due to our splitting in the AES-CTR encryption and GHASH computation for generating the content ciphertext and corresponding tag, we need only really consider the performance of the GHASH function, shown below.

[1] The design can easily be amended to use a different symmetric key and counter IV for interest and content encryption.

Table 2: Published performance measurements for the cryptographic algorithms to be used in the implementation of ANDāNA-v2. AES performance on Intel processors is far superior to other block ciphers with the AES-NI instruction set. We also note that it was not our intention to find a set of performance measurements for each algorithm on the same machine. The data in this table simply serves as a reference to the relative difference in cost for each algorithm.

Primitive	Platform/Specifications	Cycles/Byte
AES-GCM [6]	Sandy Bridge	2.53
AES-GCM [6]	Haswell	1.03
AES-GCM [6]	i7-2600K Processor (no AES-NI*)	10.42
AES-HMAC [6]	Sandy Bridge	6.14
AES-HMAC [6]	Haswell	4.71
GHASH [6]	Sandy Bridge	1.79
GHASH [6]	Haswell	0.40
HMAC	*	*
SHA-256 [4]	i3-2310M; h6sandy; supercop-20120521	17.25
Keccak-256 [4]	i3-2310M; h6sandy; supercop-20120521	10.25
SipHash-2-4 (long messages) [2] [5]	amd64 64-bit	1.95
SipHash-2-4 (long messages) [5]	i3-2310M; Sandy Bridge (206a7)	2.98

Table 3: Cryptographic algorithm performance measurements based on the current (i.e., latest stable release) OpenSSL implementation. All metrics were captured on a system running Ubuntu 13.04 32-bit with an Intel Core i7-3770 CPU (@ 3.4Ghz x 8) with 16GB DDR3 memory.

Primitive	Cycles/Byte
AES-128 (256 size blocks)	192.58
AES-128 (1024 size blocks)	189.94
GHASH (256 size blocks)	15.62
GHASH (1024 size blocks)	15.17
HMAC (256 size blocks)	75.659
HMAC (1024 size blocks)	47.2947
SHA-256 (256 size blocks)	146.727
SHA-256 (1024 size blocks)	119.738
Keccak-256 (256 size blocks)	42720.00
Keccak-256 (1024 size blocks)	192362.00
SipHash-2-4 (256-bit messages)	1.65
SipHash-2-4 (1024-bit messages)	1.78

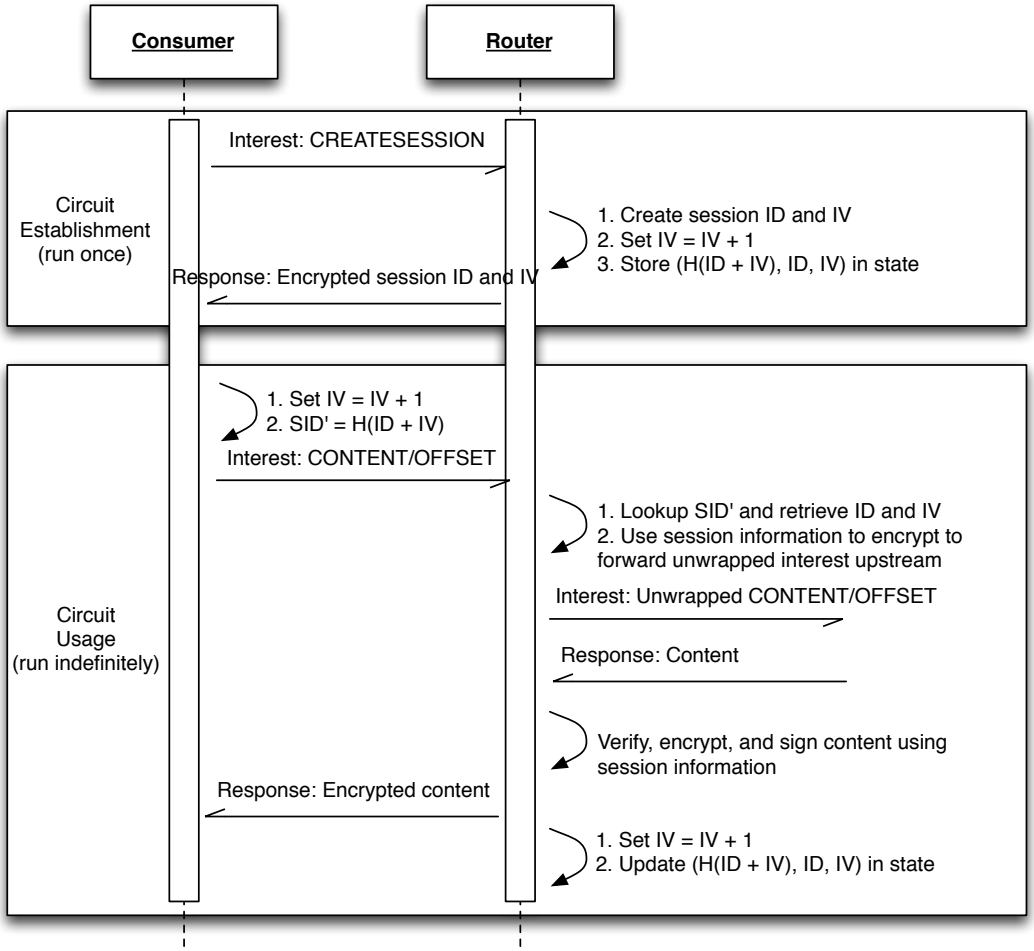

Figure 3: Visual depiction of the interaction between the consumer and the first hop router. The procedure repeats in the same manner for all further upstream routers after each interest is unrolled and resulting content is encrypted.

$$X_i = \begin{cases} 0 & \text{for } i = 0 \\ (X_{i-1} \oplus A_i) \cdot H & \text{for } i = 1, \ldots, m-1 \\ (X_{m-1} \oplus (A_m^* || 0^{128-v})) \cdot H & \text{for } i = m \\ (X_{i-1} \oplus C_{i-m}) \cdot H & \text{for } i = m+1, \ldots, m+n-1 \\ (X_{m+n-1} \oplus (C_n^* || 0^{128-u})) \cdot H & \text{for } i = m+n \\ (X_{m+n} \oplus (\mathsf{len}(A) || \mathsf{len}(C))) \cdot H & \text{for } i = m+n+1 \end{cases}$$

Note that all multiplication $\cdot$ is done over polynomials in $GF(2^{128})$, where H is the ciphertext corresponding to the plaintext 0^n. In our design there will be no additional authenticate data (AAD); perhaps the session identifier could be used, but we do not require nor enforce any AAD. Fortunately, there exists numerous techniques to speed up the performance of this procedure, including carry-less Karatsuba multiplication and (deferred) polynomial reduction via the "Montgomery" technique [6]. Collectively, it is clear that performance of the GHASH computation is expected to exceed performance of HMAC(-SHA-1).

As for the Keccak hash function, we do not expect to achieve any significant performance improvements beyond what is already presented in the literature. We will use the optimized implementation provided by

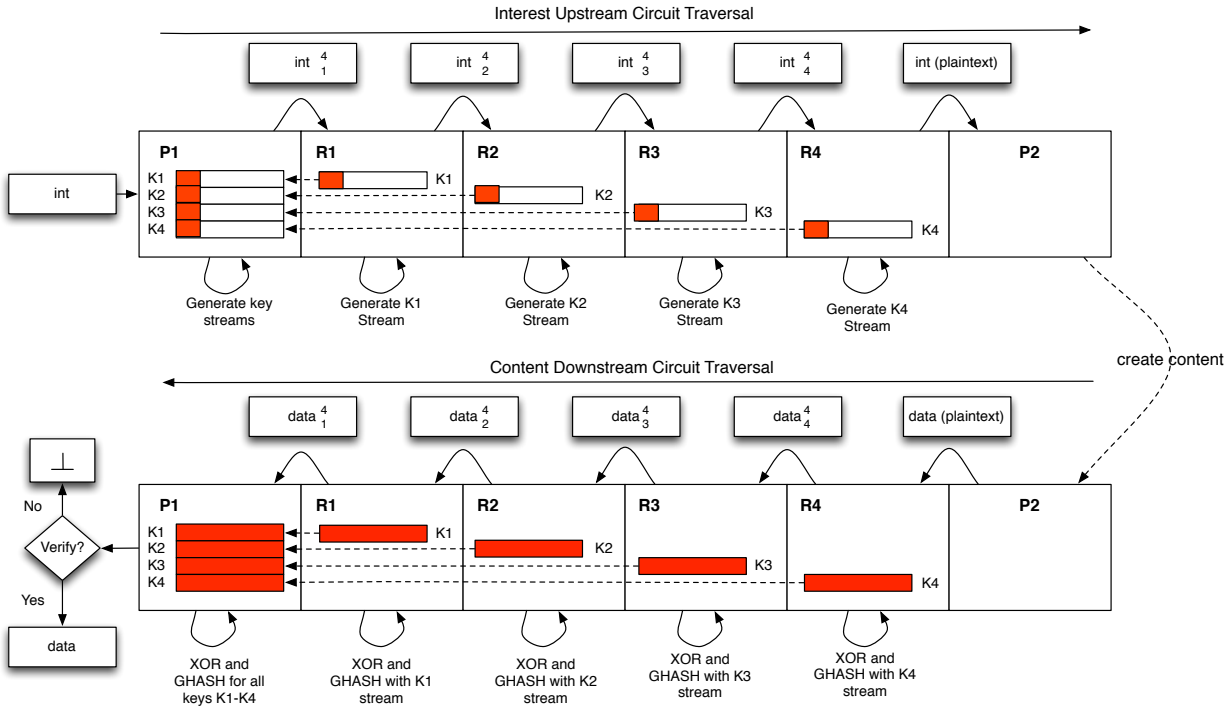

Figure 4: Visual depiction of keystream precomputation during upstream interest traversal and the resulting half of AES-GCM that must be computed for the resulting content flowing downstream. We again emphasize that keystream "pumping" for AES-GCM can be done for interests as well.

the authors in the SHA-3 competition [10]. Similarly, due to the high performance of the optimized SipHash procedure, we will not explore any additional optimizations for this algorithm.

5 Preliminary Performance Evaluation and Cost Models

In order to justfiy ANDāNA-v2 as an alternative to existing solutions for low-latency, bidirectional traffic over NDN, we must first establish a baseline of performance metrics against which we can compare our design. Since the original version of ANDāNA targeted circuits of length 2 (i.e., with two ARs), we focus on the same length circuits in these experiments. We note that there is nothing restricting us from lifting this constraint in future experiments. In order to adequately determine a baseline of performance metrics, we instantiated two parties (both acting as a data producer and consumer) interacting by requesting moderately-sized content in short, frequent intervals. In order ensure that such content is never satisfied by the cache of an intervening AR, which is a reasonable assumption for real-time voice and video applications that always want fresh content instead of stale cached content, each piece of content is requested from an anonymized namespace and indexed by a sequence number. For example, if the two parties P_1 and P_2 are communicating and P_1 wants to request the latest content from P_2, it will issue an (encrypted)interest to `ccnx:/`P_2`/`S, where S is the sequence number that is incremeneted as soon as the interest is issued. This ensures that such interests are never satisfied by the content in an AR cache throughout the duration of the session.

With this type of traffic generation application, we then tested its performance under the following scenarios:

1. P_1 and P_2 are connected point-to-point (i.e., no hops).
2. P_1 and P_2 are connected through two "insecure" ARs that perform no interest or content encryption (i.e., each AR just serves as an application-level proxy to forward interests and content along through

Table 4: Baseline performance metrics gathered from the three scenarios discussed in the text.

Scenario	L_1	σ_1	L_2	σ_2
1	0.00735	0.00798	0.00340	0.00053
2	0.01905	0.13791	0.02248	0.13923
3	0.01321	0.00257	0.01359	0.00567

the circuit).

3. P_1 and P_2 are connected through two "secure" ARs that perform interest and content encryption as per the ANDāNA design.

Table 4 shows the performance results from such scenarios 1, 2, and 3, in which performance is characterized as the end-user latency between issuing an interest and receiving the intended content. In this case, we refer to the performance in terms of latency, where L_i denote the average latency and σ_i denote its standard deviation for party i ($i \in \{1, 2\}$). All experiments were collected using the following methdology:

- Each party generates 1000 traffic according to an uniform distribution with $pmf = 1/100$.
- Each party responds to all interests with a random blob of 150 bytes. [3]

According to the preliminary experimental evaluation, it appears as though low-latency, bidirectional communication is realtively feasible with the current ANDāNA design. However, we would like to do better. To that end, we first formally capture the performance of the ANDāNA (symmetric variant) and ANDāNA-v2 designs to enable quantitative comparison using the concrete cryptographic primitives discussed in the previous following section. First, let T_1^{S} and T_2^{A} be the expected total time required to retrieve one piece of content in the symmetric ANDāNA and asymmetric ANDāNA-v2 designs, respectively. Assume that a circuit consists of n ARs, all encrypted interests consist of a single 128-bit block, and all respective content packets consist of b 128-bit blocks. Finally, let T_{AES}, T_{HMAC}, T_{GHASH}, T_{RSA}, T_{MAC}, and T_{HASH} denote the time required to perform AES, HMAC, GHASH, RSA, MAC, and an arbitrary hash function computation, respectively. Note that we separate the time required to encrypt a single block of plaintext with AES from the corresponding MAC scheme (i.e., HMAC or GHASH) for notational convenience. We also introduce the additional quantities $H_1 = \sum_{i=0}^{n-1} i$ and $H_b = \sum_{i=0}^{n-1}(b+i)$, which will be used to count the blocks of ciphertext that are accumulated as interests and content flow upstream and downstream. We now present the expressions for T_1^{S} and T_2^{A} as follows:

$$T_1^{\mathsf{S}} = \underbrace{2n(T_{\mathsf{AES}} + T_{\mathsf{HMAC}})}_{\text{interests}} + \underbrace{2nb(T_{\mathsf{AES}} + T_{\mathsf{HMAC}})}_{\text{content encryption}} + \underbrace{2T_{\mathsf{HASH}}\left(nb + \frac{n(n+1)}{2}\right) + 2nT_{\mathsf{RSA}}}_{\text{sign/verify}}$$

$$T_2^{\mathsf{A}} = \underbrace{2n(T_{\mathsf{HASH}} + T_{\mathsf{AES}} + T_{\mathsf{GHASH}})}_{\text{interest encr/decr}} + \underbrace{2n(b+1)T_{\mathsf{GHASH}}}_{\text{content encr/decr}} + \underbrace{2T_{\mathsf{HASH}}\left(nb + \frac{n(n+1)}{2}\right) + 2nT_{\mathsf{MAC}}}_{\text{sign/verify}}$$

It is important to emphasize that we are comparing the symmetric variant of ANDāNA with the asymmetric variant of ANDāNA-v2. As stated in [2], the symmetric variant does not enjoy consumer and producer unlinkability since session identifiers are sent in cleartext. However, for the sake of comparing performance, we opted to compare the *best* version of ANDāNA with the *worst* variant of ANDāNA-v2. Accordingly, comparing the performance of these two seemingly uncompatible designs will serve to intensify the improvement of the ANDāNA-v2 design both in terms of performance (for this particular use case) and security/anonymity. As one can see, T_1^{S} and T_2^{A} are quite different. While content signature generation seemingly consumes the

[3] This value was chosen to match the average size of a Skype video call packet. Are experiments with different size pieces of content warranted for this proof of concept?

majority of the computation in the ANDāNA design, the ANDāNA-v2 design enjoys highly efficient content signature generation by using symmetric MACs. Since adjacent ARs in the circuit share a common MAC key that is established during the circuit and session establishment protocol, a MAC can effectively remove the need for an expensive hash computation (on possibly large messages) and subsequent signature. Furthermore, there are no asmmetric cryptographic operations needed for this new design, which intuitively means that it will achieve better performance. Of course, we require empirical evidence to support this claim, and the implementation and evaluation of ANDāNA-v2 will provide exactly that information.

6 Correctness and 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 [2]:

- Deploy compromised routers
- Compromise existing routers
- Control content producers
- Deploy compromised caches
- 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 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 [2], 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 [2], 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

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

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 [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 ANDāNA-v2 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 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 [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 $u \in (\mathsf{C} \setminus \mathsf{C}_{\mathcal{A}})$ 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 $\Pr[\mathsf{MacForce}_{\mathcal{A},\Pi}(1^\kappa) = 1] \leq \mathsf{negl}(\kappa)$, 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 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.[4] □

References

[1] V. Jacobson, D. K. Smetters, N. H. Briggs, M. F. Plass, P. Stewart, J. D. Thornton, and R. L. Braynard. VoCCN: Voice-Over Content-Centric Networks. *In Proceedings of the 2009 Workshop on Re-Architecting the Internet (ReArch '09), ACM, New York, NY, USA* (2009), 1-6.

[2] S. DiBenedetto, P. Gasti, G. Tsudik, and E. Unzun. ANDāNA: Anonymous named data networking application. *In NDSS '12* (2012).

[3] R. Dingledine, N. Mathewsonn, and P. Syverson. Tor: The Second-Generation Onion Router. *In The 13th USENIX Security Symposium* (2004).

[4] S. Chang, R. Perlner, W. E. Burr, M. S. Turan, J. M. Kelsey, S. Paul, L. E. Bassham. Third-Round Report of the SHA-3 Cryptographic Hash Algorithm Competition. *National Institute of Standards and Technology (NIST) NISTIR 7896.*

[5] J. Aumasson and D. J. Bernstein. SipHash: A Fast Short-Input PRF. *Progress in Cryptology - INDOCRYPT 2012, Springer Berlin Heidelberg* (2012), 489-508.

[6] S. Gueron. AES-GCM for Efficient Authenticated Encryption - Ending the Reign of HMAC-SHA-1? *Workshop on Real-World Cryptography, Stanford University, CA* (2013).

[7] M. Franz, B. Meyer, and A. Pashalidis. Attacking Unlinkability: The Importance of Context. *Privacy Enhancing Technologies. Springer Berlin Heidelberg* (2007).

[8] S. Schiffner and S. ClauB. Using Linkability Information to Attack Mix-Based Anonymity Services. *Privacy Enhancing Technologies. Springer Berlin Heidelberg* (2009).

[9] S. J. Murdoch and G. Danezis. Low-Cost Traffic Analysis of Tor. *2005 IEEE Symposium on Security and Privacy* (2005).

[10] The Keccak sponge function family. Software and other files. Available online at `http://keccak.noekeon.org/files.html`. Last accessed 11/18/13.

[4] 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.