Network-Layer Trust in Named-Data Networking

Cesar Ghali Gene Tsudik
University of California, Irvine
{cghali,gene.tsudik}@uci.edu

Ersin Uzun
Palo Alto Research Center
Ersin.Uzun@parc.com

ABSTRACT

In contrast to today's IP-based host-oriented Internet architecture, Information-Centric Networking (ICN) emphasizes content by making it directly addressable and routable. Named Data Networking (NDN) architecture is an instance of ICN that is being developed as a candidate next-generation Internet architecture. By opportunistically caching content within the network, NDN appears to be well-suited for large-scale content distribution and for meeting the needs of increasingly mobile and bandwidth-hungry applications that dominate today's Internet.

One key feature of NDN is the requirement for each content object to be digitally signed by its producer. Thus, NDN should be, in principle, immune to distributing fake (aka "poisoned") content. However, in practice, this poses two challenges for detecting fake content in NDN routers: (1) overhead due to signature verification and certificate chain traversal, and (2) lack of trust context, i.e., determining which public keys are trusted to verify which content. Because of these issues, NDN does not force routers to verify content signatures, which makes the architecture susceptible to content poisoning attacks.

This paper explores root causes of, and some cures for, content poisoning attacks in NDN. In the process, it becomes apparent that meaningful mitigation of content poisoning is contingent upon a network-layer trust management architecture, elements of which we construct, while carefully justifying specific design choices. This work represents the initial effort towards comprehensive trust management for NDN.

1. INTRODUCTION

The Internet usage model has changed considerably over the last two decades. Limitations of the current Internet are becoming more pronounced as network services and applications become increasingly mobile and data-centric. In recent years, a number of research efforts have sprung up aiming to design the next-generation Internet architecture. Some are based on the notion of Information-Centric Networking (ICN) which emphasizes efficient and scalable content distribution. Named Data Networking (NDN)[24], a fork from PARC's Content Centric Networking (CCNx) architecture[5], is one such research effort. One of the main tenets of NDN is named content. NDN also stipulates in-network content caching, by routers. To secure each content, NDN requires it to be cryptographically signed by its producer. This way, globally addressable and routable content can be authenticated by anyone, which allows NDN to decouple trust in content from trust in entities that store and disseminate it. NDN entities that request content are called *consumers*. A consumer is expected to verify content signatures in order to assert:

- *Integrity* – a valid signature (computed over a content hash) guarantees that signed content is intact;
- *Origin Authentication* – since a signature is bound to the public key of the signer, anyone can verify whether content originates with its claimed producer;
- *Correctness* – since a signature binds content name to its payload, a consumer can securely determine whether delivered content corresponds to what was requested;

Although any NDN entity can verify any content signature, NDN routers are *not required* to do so. This is not only because of the overhead stemming from the actual cryptographic verification of the signature itself. There are two other, more important, reasons for not mandating router verification of content signatures:

1. First, a router must be aware of the specific trust model for each content-producing application. Given the wide range of possible applications, it is very unlikely that they will all use the same trust model. Some applications will probably use trust hierarchies, while others might adopt a flat peer-based trust models, or hybrid versions thereof. Furthermore, the set of NDN applications will change over time. Also, the trust model of a particular application might not be static in the long term.
2. Second, depending on the trust model of an application associated with a particular content, a router needs access to – and thus might need to fetch[1] – multiple public key certificates or similar structures in order to trust the public key that verifies a content signature. For example, if an application uses a hierarchical PKI, an entire root-to-leaf path might have to be traversed and all intermediate certificates would need to be separately verified. This would need to include ancillary activities for each such certificate, i.e., expiration and revocation checking.

These issues greatly complicate network-layer trust management in NDN. One easy alternative – adopted by the current version of NDN – is to make it optional for routers to verify content signatures. Unfortunately, this decision leaves NDN vulnerable to content poisoning attacks on router caches. To make matters worse, NDN does not provide any definitive mechanism for a consumer to request genuine desired content. Instead, a consumer that receives fake content can explicitly exclude the latter (by referring to its hash) in subsequent requests. This does not guarantee eventual success, due to the potentially unbounded number of fake content objects sharing the same name.

This undesirable state-of-affairs serves as the main motivation for our work. In this paper, we analyze NDN architecture and its susceptibility to content poisoning attacks. Next, we postulate some intuitive goals for routers to support trust management and content validation. We then present simple rules that allow NDN parties (consumers, producers and routers) to mitigate content poisoning, while minimizing trust-related complexity for routers. These

[1]The alternative of carrying the entire collection of certificates as part of each content is clearly undesirable.

rules require no changes to the fundamentals of the NDN architecture.

Besides being the first effort to address content poisoning and trust management in NDN, one contribution of this work is in careful analysis and justifications for placement and complexity of various trust mechanisms.

Disclaimer: it is impossible to predict whether NDN will ever cross the line between a research prototype and a widely deployed architecture. NDN, similar to every other candidate Future Internet Architecture, has its benefits and pitfalls. The purpose of this work is not to advocate for or against NDN. Instead, we aim to improve NDN security features by utilizing techniques in [27, 14, 15, 10, 19]. Our main goal is to provide NDN routers with a mechanism to efficiently and securely verify content in order to mitigate content poisoning attacks.

Scope: as reflected in the title, this paper focuses on network-layer trust issues, motivated by the content poisoning problem. We do not address other NDN security issues, such as interest flooding attacks [7, 12, 2], cache pollution attacks [9] and routing security [29].

2. NDN OVERVIEW

Unlike IP which focuses on end-points of communication and their names/addresses, NDN ([18, 24]) emphasizes content and makes it named, addressable and routable at the network layer. A content name is composed of one or more variable-length components opaque to the network. Component boundaries are explicitly delimited by "`/`" in the usual path-like representation. For example, the name of CNN home-page content for August 20, 2014 might be: `/ndn/cnn/news/2014august20/index.htm`. Large content can be split into segments with different names, e.g., fragment 37 of Alice's YouTube video could be named:
`/ndn/youtube/alice/video-749.avi/37`.

NDN communication adheres to the *pull* model and content is delivered to consumers only following an explicit request. There are two types of packets in NDN: interest and content. A consumer requests content by issuing an *interest* packet. If an entity can "satisfy" a given interest, it returns a corresponding *content* packet. Content delivery must be preceded by an interest. If content C with name n is received by a router with no pending interest for n, C is considered unsolicited and is discarded. Name matching in NDN is prefix-based. For example, an interest for `/ndn/youtube/alice/video-749.avi` can be satisfied by content named `/ndn/youtube/alice/video-749.avi/37`.[2]

NDN content includes several fields. In this paper, we are only interested in the following:

- `Signature` – a public key signature, generated by the content producer, covering the entire content, including all explicit components of the name and a reference to the public key needed to verify it.
- `Name` – a sequence of explicit name components followed by an implicit `digest` (cryptographic hash) component of the content that is recomputed at every hop. This effectively provides each content with a unique name and guarantees a match with a name provided in an interest. However, in most cases, the `digest` component is not present in interest packets, since NDN does not provide any secure mechanism for a consumer to learn a content hash, prior to requesting it.
- `PublisherPublicKeyDigest (PPKD)` – SHA-256 digest of the public key needed to verify the content signature.
- `Type` – content type, e.g., data, encrypted content, key, etc.
- `Freshness` – recommended content lifetime (after being cached) set by the producer.
- `KeyLocator` – reference to the public key required to verify the signature. This field has three options: (1) verification key, (2) certificate containing the verification key, or (3) NDN name referencing the content that contains the verification key.

Each content producer must have at least one public key, represented as a *bona fide* named content of `Type` $=$ *key*, signed by its issuer, e.g., a certification authority (CA).[3] The naming convention for a public key content object is to contain "key" as its last explicit component, e.g., `/ndn/russia/moscow-airport/transit/snowden/key`.

An NDN interest includes the following fields:

- `Name` – NDN name of requested content.
- `Exclude` – contains information about name components that **must not** occur in the name of returned content. This field can also be used to exclude certain content by referring to its `digest`, which, as noted above, is included in the content as an implicit last component of each content name, or in a separate field.
- `PublisherPublicKeyDigest (PPKD)` – the SHA-256 digest of the publisher public key. If this field is present in the *interest*, a matching content objects must have the same digest in its `PPKD`.

There are three types of NDN entities[4]: (1) *consumer* – an entity that issues an interest for content, (2) *producer* – an entity that produces and publishes (as well as signs) content, and (3) *router* – an entity that routes interest packets and forwards corresponding content packets. Each entity (not just routers) maintains the following three components:

- *Content Store* (CS) – cache used for content caching and retrieval. From here on, we use the terms *CS* and *cache* interchangeably.
- *Forwarding Interest Base* (FIB) – routing table of name prefixes and corresponding outgoing interfaces used to route interests. NDN does not specify or mandate any routing protocol. Forwarding is done via longest-prefix match on names.
- *Pending Interest Table* (PIT) – table of outstanding (pending) interests and a set of corresponding incoming and outgoing interfaces.

When a router receives an interest for content named n which is not in its cache, and there are no pending interests for the same name in its PIT, it forwards the interest to the next hop(s), according to its FIB. For each forwarded interest, a router stores some amount of state information, including the name in the interest and the interface on which it arrived. However, if an interest for n arrives while there is a pending entry for the same content name in the PIT, the router collapses the present interest (and any subsequent interests for n) storing only the interface upon which it was received. If and when content is returned, the router forwards it out on all incoming-interest interfaces and flushes the corresponding PIT entry. Since no additional information is needed to deliver content, an interest does not carry any *source address*. If a content fails to arrive before some router-determined expiration time, the router can either flush the PIT entry or attempt interest re-transmission over the same, or different, interfaces.

An NDN router's cache size is determined by local resource availability. Each router unilaterally determines which content to cache and for how long, though lifetime (as mentioned above) can be recommended by the producer. Upon receiving an interest, a router first checks its cache to see if it already has requested content in its cache. Producer-originated content signatures allow con-

[2]The reverse does not hold, by design.

[3]Recall that NDN is agnostic as far as trust management, aiming to accommodate peer-based, hierarchical and hybrid PKI approaches.

[4]Note that a physical entity (a host, in today's parlance) can be both consumer and producer of content.

sumers and routers to authenticate received content, regardless of the entity serving it.

3. CONTENT POISONING

The central objective of NDN is efficient and scalable distribution of information. This is facilitated by routers opportunistically caching content. Whenever an NDN router receives an interest for a name that matches a content in its cache, it satisfies the interest with that content. Since routers are not required to verify signatures, the delivered content is not guaranteed to be authentic. However, a consumer is required to verify signatures of all returned content. A consumer is thus assumed to have the necessary application-specific trust context to decide which public keys to trust. This allows consumers to reliably detect fake content.

However, NDN offers no means for consumers to ask routers to *flush* fake content from their caches. The only recourse for a consumer that detects fake content is to issue another interest that specifically excludes the unwanted content by specifying its hash in the exclusion filter field of the new interest. Unfortunately, this explicit exclusion does not signify (to routers) bad or poisoned content, as the same feature can also be used to exclude stale content. Furthermore, even if the exclusion technique were to be used strictly for flagging poisoned content, the result would be undesirable, for the following reasons:

The entire notion of consumers (i.e, end-systems or hosts) informing routers about poisoned content is full of pitfalls. Suppose a consumer complains to a router about specific content. If this is done without consumer authentication (whether via an interest, e.g, using exclusion, or via a separate packet type), the router would have two choices: (1) immediately flush referenced content from its cache, or (2) verify the content signature and flush content only if verification fails. The former (1) is problematic, since it opens the door for anyone to cause easy removal of popular content from router caches, which can be considered as a type of a denial-of-service attack. Even if this were not an issue, there would remain a more general problem: as noted in [12], the adversary mounting a content poisoning attack could continue *ad infinitum* to feed new invalid content in response to interests that exclude previously consumer-detected invalid content. The second option (2) is also problematic, because, besides the cost of verifying a signature (which can lead to a denial-of-service attack by itself), it brings back the problem of routers having to understand potentially complex trust semantics of many diverse content-producing applications.

Another possibility is to require consumers to authenticate themselves when complaining about poisoned content. This would entail signing the interest (or another new packet type) that complains about allegedly bad content. One unpleasant privacy consequence is that the signer (consumer) would be exposed by the signature, since it would need to be bound to a public key, contained in a certificate. (This certificate would have to be communicated with each complaint message, along with auxiliary information that the router would need to trust the certificate.) More generally, signing would violate one of the key elements of NDN architecture – consumer opacity. Recall that producers sign content, while consumers do not sign interests, or any other messages.

Another reason why consumer signing of "complaint" messages is problematic is because it can be abused to mount DoS attacks on routers by flooding them with junk complaints and forcing expensive signature verification.[5] Note that, even if the router successfully authenticates a consumer complaint, this is no guarantee that the accused content is fake; in order to be sure, the router would have to verify the content signature as well. Moreover, authentication of consumers by routers would require identity management and verification systems to be in place at the network layer, thus adding significant overhead.

Finally, the preceding discussion applies not only to content *cached* by routers. Since NDN only recommends, and does not mandate, content caching, it is entirely *legal* for a router not to cache some, or all, content that it forwards. If a router does not cache C, then complaining about C being fake is clearly useless.

At this point, it becomes clear that dealing with fake content represents a real challenge for NDN. Although some light-weight non-cryptographic and partially effective counter-measures have been proposed (e.g., [13]), they do not fully address the problem and quickly become ineffective against an active adversary.

3.1 Zooming In

Based on the above arguments and recent results simulating content-poisoning attacks [13], we conclude that NDN has a major security problem, since it offers: (1) no way to prevent fake content from being delivered to consumers, and (2) no way to reliably flush invalid content from router caches. There are two reasons for this:

1. Ambiguous interests: NDN requires each interest to carry the name of desired content. However, neither the `digest` component of the name, nor the `PPKD` is a required field in an interest. In other words, an interest for a content name can be satisfied by multiple content objects, including those with untrusted or unverifiable signatures.

2. No unified trust model: even if routers could verify signatures at line speed, NDN does not provide a trust model enforceable at the network layer. Although two aforementioned selector fields can be used to communicate content-specific trust context to the network layer, NDN has no mechanism for a consumer to securely pre-acquire the hash of a given content, or the specific public key that should be used to verify a content signature.

3.2 Goals

As a first step in addressing the content poisoning problem, it is necessary to acknowledge the obvious, i.e., that **network-layer trust management and content poisoning are inseparably conjoined**. Since content is the basic unit of network-layer "currency" in NDN, trust in content (and not in its producers or consumers) is the central issue at the network layer.

Second, trust-related complexity (activities, state maintenance, etc.) must be minimized at the network layer. Specifically, as part of validating content, **a router should not: fetch public key certificates, perform expiration and revocation checking of certificates, maintain its own collection of certificates, or be aware of trust semantics of various applications**.[6]

On a related note, we claim that, ideally, **a router should verify at most one signature per content**. This upper-bounds the heavier part of content-related cryptographic overhead; the other part is computing a content hash. Ideally, a router would not perform any signature verification at all. However, as discussed below, this might be possible for some, yet not all, content. Also, although verifying a signature given an appropriate public key is a mechanical operation, a router would still need to support multiple signature algorithms since uniformity across all applications is improbable.

The above discussion implies that NDN entities other than routers, i.e., **producers and consumers of content, should bear the brunt of trust management**.

[5] The same attack does not work with flooding of routers with junk content since content can not be sent unsolicited and a router would only attempt signature verification of incoming content for which it has a pending interest entry in its PIT.

[6] This is separate from trust management for routing protocols.

4. THE INTEREST-KEY BINDING RULE

Ghodsi et al. [14] informally argue that, for each content, at least two out of three possible bindings (producer-key, name-key, producer-name) must be present. The third binding is transitively inherited from the other two. Due to the use of human-readable names in NDN, producer–name binding can be easily inferred.[7] Our approach to network-layer trust adheres to all goals outlined above. It is based on the binding between a name and the public key used to verify the content signature. We denote it as the *Interest-Key Binding (IKB)* rule:

IKB: An interest must reflect the public key of the producer.

A very similar concept – *self-certifying naming scheme* – is described in [15]. As discussed in Section 5, this concept needs to be adjusted for the NDN context. Recall that NDN interest format (Section 2) includes an optional field `PPKD` which serves exactly this purpose. Our approach makes it mandatory without any substantive changes to the NDN architecture.

An NDN public key is a special type of content in the form of a certificate signed by the issuing CA. Each certificate contains a list of all name prefixes that it is authorized to sign/verify. The name of the certificate-issuing (content-signing) CA and the name of the key contained in a certificate (content) are not required to have any specific relationship. This is part and parcel of NDN's philosophy of leaving trust management up to the application, e.g., signed content C can be verified with public key PK with C and PK having no common prefix. For instance, content containing the public key `/cnn/usa/web/key` could be issued and verified by the key `/verisign/key`. Of course, an application is free to impose all kinds of restrictions, as long as routers remain oblivious.

4.1 Implications for Producers and Routers

We now examine IKB implications on content producers and routers, respectively.

For content producers, IKB has very few consequences. In fact, it simplifies content construction by asking the producer to include the public key itself in the `KeyLocator` field of content. In other words, IKB obviates two other current NDN options: (1) referring to a verification key (via the `KeyLocator` field) by its name, or (2) including it in a form of a certificate.

For NDN routers, IKB implications are overwhelmingly positive. First, a router needs to perform no fetching, storing or parsing of public key certificates, as well as no revocation or expiration checking. All such activities are left to consumers.

Upon receiving a content and identifying the PIT entry corresponding to one or more pending interests a router simply hashes the public key from the content `KeyLocator` field and checks whether it matches the `PPKD` of the PIT entry. In case of a mismatch, the content is discarded.[8] Otherwise, the content signature is verified and (if valid) the content is forwarded and cached.

The implications would be even more beneficial for producers and routers with the use of self-certifying content names (SCNs), as discussed in Section 5 below. With this optimization, inclusion of key information and signature checking could be avoided for most content objects, thus further reducing the communication and computation overhead.

4.2 Implications for Consumers

For consumers, IKB does not increase complexity. It actually prompts us to codify desired consumer behavior – something that has been left unspecified in the NDN architecture.

[7]If we assume that names are clear and unambiguous.

[8]A slightly simpler alternative is to perform PIT lookup each incoming content by using both content name and public key hash.

The most immediate IKB consequence for a consumer is the need to **obtain and validate the producer's public key before issuing an interest for any content originated by that producer.** At the first glance, this might appear to be an example of the proverbial "chicken-and-egg" problem. However, we show below that this is not the case.

A consumer that wants to fetch certain content C is doing so as part of some NDN application, APP_C. We assume that a consumer must have already installed this application. APP_C must have a well-defined trust management architecture that is handled by its consumer-side software. However, the remaining question is: how to bootstrap trust and how to obtain initial public keys?

We consider three non-exclusive alternatives:

(1) One possibility is that APP_C client-side software comes with some pre-installed root public key(s), perhaps contained within self-signed certificates. Without loss of generality, we assume that there is only one such key – PK_{root}. Armed with it, a consumer can request lower-level certificates, by issuing an interest referencing the hash of PK_{root} in the `PPKD` field.[9]

(2) Alternatively, one could imagine a global Key Name Service (KNS), somewhat akin to today's Domain Name Service (DNS). In response to consumer-issued interests referencing public key names and/or name prefixes, KNS would reply with signed content containing one or more public key certificates (i.e., as embedded content) corresponding to requested names.

(3) A similar approach is a global search-based service, i.e., something resembling today's Google. A consumer would issue a search query (via an interest) to the search engine which would reply with signed content representing a set (e.g., one page at a time) of query results. One or more of those results would point to content corresponding to the public key certificate of interest to the consumer.

In cases (2) and (3), consumers would still need to somehow securely obtain the root public keys for KNS and the search engine, respectively. This can be easily done via (1).

4.3 Security Arguments

We now return to the original motivation for this work – mitigation of content poisoning attacks. We need to show that global adherence to the IKB rule leads to security against content poisoning.[10]

If we assume that:

1. Every router abides by the IKB rule and acts as described in Section 4.1.
2. Every consumer abides by the IKB rule and acts as described in Section 4.2.
3. The consumer requesting content C is not malicious.
4. Each router R that is one hop away from the consumer is not compromised.
5. The links between a consumer and its adjacent routers are not compromised.

We can briefly argue security by contradiction: Suppose that a consumer receives fake content C from R. Let Int denote the interest issued earlier by that consumer that was satisfied by C. According to IKB, Int must contain the digest of a public key of producer P in its `PPKD` field. Let PK denote this public key. Consequently, R must have made sure that: (1) C is signed with a public key PK' with a hash matching `PPKD` of Int, meaning that $H(PK') = H(PK)$ and (2) the signature itself is correct, i.e., valid. Also, since R is not malicious and all communication between R and (also not malicious) the consumer is secure, the only

[9]If APP_C comes with several root public keys, the consumer would need to issue multiple simultaneous interests referencing the hash of each root key in `PPKD`.

[10]Note that, as mentioned in Section 1, cache pollution attacks are an entirely different matter.

remaining possibility is a hash collision, i.e., $PK' \neq PK$ while $H(PK') = H(PK)$. The latter is assumed to occur with negligible probability.

This does not yet conclude our security discussion. As noted in [12], content poisoning attacks can originate with malicious routers. What happens if a malicious router R' feeds poisoned content C' to its non-malicious next hop neighbor R, towards some consumer(s)? Since R is honest and implements IKB, before forwarding and (optionally) caching C', it verifies, as before, that the signature of C' is successfully verifiable using PK that matches the hash in the corresponding PIT entry, i.e., the value of the `PPKD` field of the original interest Int that triggered creation of this PIT entry.

5. OPTIMIZATIONS

As mentioned before, IKB rule implies that routers should perform only one signature verification using the public key provided (by the producer) in the content and specified (by the consumer) using the `PPKD` field in the interest. Instead of including the public key in the content, it could be directly included by the consumer in the interest. This would require storing the public key alongside the interest in the PIT entry, to be used later for signature verification of the content. Since it is fair to assume that cache entries have longer lifetime than PIT entries, this approach can be beneficial in terms of storage. Its main drawback, however, is that the current interest format would need to be modified to include public keys.

For backbone routers that process and forward tens of gigabits per second, performing even a single signature verification per packet imposes a huge overhead. One approach to overcome this problem is to take advantage of the network structure. The current Internet is divided into Autonomous Systems (AS-s), each representing an administrative entity. In this architecture, only border routers of consumer-facing AS-s might implement the IKB rule by verifying signatures of all received contents. Alternatively, each router in an AS might probabilistically verify signatures on a subset of packets it forwards. The drawback of these approaches are that fake content could still be cached by routers that did not verify its signature. However, either method would have good chance of detecting and discarding most fake content before reaching to the consumer.

Another way to reduce signature verification overhead is to use SCNs [15, 11, 22, 12, 3]. According to [28], a content name can only have at most two out of the following three properties: security, uniqueness and human-readability. As suggested in [14, 15], SCNs can be formed by appending to the producer's public key digest a label that uniquely identifies the content. While this approach guarantees security and uniqueness, it lacks the means of verifying the binding between the content and its name [27]. To overcome this issue in NDN, we consider forming an SCN by specifying the hash of requested content as the last component of the content name in the interest [17]. This provides name-content as well as producer-name, producer-key and name-key security bindings (as in Section 4). Although this use of SCNs does not yield fully human-readable names, it provides uniqueness and security properties [14].

If a benign NDN consumer uses SCNs, the network guarantees (due to longest-prefix matching) delivery of "valid" content. The main advantage is that routers no longer need to verify signatures. Instead, they only recompute a content hash and check that it matches the one in the corresponding PIT entry. The remaining question is: how can a consumer obtain the hash of a content beforehand?

For the type of communication where most content is requested using SCNs, we advocate the use of so-called *catalogs*. A catalog is basically an authenticated data structure that includes a set of SCNs. This set can consists of references to content objects containing data, public keys, or other catalogs. The structure of catalogs can be application-specific and might vary from a simple list of SCNs, to multiple SCN sets forming a Merkle tree [23] or some similar data structure. To securely fetch an initial catalog, a consumer can fall back to using the `PPKD` interest field, as discussed earlier.

One obvious corollary of using SCNs in interest messages is that consumers and routers are no longer required to verify content signatures, as long as the SCN is trusted, i.e., obtained from a (consumer-verified) catalog. This reduces: (1) overhead of publishing, since producers now sign catalogs rather than individual content, and (2) network overhead, since there is no need to add the public key to the `KeyLocator` field of the content, as discussed in Section 4. The only time a signature is required is whenever a content is requested via `PPKD` interest field. In that case, both routers (prior to serving content from cache or forwarding it) and consumers (prior to accepting) *must* perform content signature verification. We believe that is should be left up to the producer to decide whether a content should be requested by specifying its corresponding public key, SCN, or both.

Using SCNs in conjunction with catalogs brings up the issue of unsigned content objects. In other words, a content C which is indirectly signed as part of a catalog, can be fetched by its SCN, i.e., name-hash combination. This does not rule out C being separately signed by its producer. However, signing a catalog-ed content increases overhead for the producer and increases content size. A sensible approach is not to sign catalog-ed content objects at all. This would imply that such objects can *only* be fetched via SCN. However, NDN architecture requires each content to be *individually* verifiable. Thus, existence of unsigned objects conflicts with a basic tenet of NDN [11].

6. PROPOSED MODEL IN PRACTICE

NDN was designed as a candidate next-generation Internet architecture. In order to provide a smooth and successful transition path, NDN must contend with application-specific requirements, such as trust. In this section we discuss how the aforementioned trust model and its optimization could be applied in practice. We start by identifying different traffic types.

6.1 Content Distribution

This type of traffic corresponds to client-server communication in and accounts for well over 90% of current Internet traffic[16]. Since most requested content is static, creating secure catalogs is straightforward. Consumers request catalogs and then use included SCNs to request desired content. We consider two common examples of content distribution traffic:

<u>Audio/Video Streaming:</u> A typical audio/video is a large content split into several segments with different names (as mentioned in Section 2). If a catalog containing the SCNs of all the segments can be provided, consumers can use these names in subsequent interests to retrieve all segments of the content.

<u>Internet Browsing:</u> We anticipate that most HTML files would fit into a single content object [26, 1]. A typical HTML file contains reference links to other static and dynamic content, such as images, audio or other HTML pages (sub-pages). While rendering HTML files, Internet browsers parse all reference links and download corresponding content. Therefore, if an HTML file uses SCNs as references, it can be viewed and treated as a secure catalog. Of course, SCNs can only be used for static content, since the hash of

[11] This is not the case for the latest version of CCNx, the original architecture that spawned NDN. CCNx 1.0 adopts secure catalogs (called manifests) and its packet format supports unsigned content objects[5].

dynamic (e.g., generated upon request) content cannot be known *a priori*.

Internet browsing provides a good example of content that can be requested via either PPKD or SCNs. Suppose that a web page A contains a reference link to sub-page B and this link is expressed using an SCN. Once a consumer requests and obtains page A, the client browser can request B using the appropriate SCN in A. Whereas, other consumers might wish to directly request page B (not as part of A) using its PPKD. Note that, for obvious reasons, SCNs can not be used with HTML pages (or any other content) with circular references, e.g., $A \leftrightarrow B$.

6.2 Interactive Traffic

Another major traffic type corresponds to interactive communication, where content is generated on demand. Applications such as voice/video conferencing, remote terminal access and on-line gaming fall into this category. Such applications generally benefit from network caching only in cases of packet loss, since re-issued interests for lost packets are likely to be satisfied by the first hop NDN router. Obviously, the use of large catalogs for interactive real-time traffic is neither sensible nor feasible. Instead, consumers should request content by using PPKD in interests, in conjunction with producers perhaps offering small dynamically-generated catalogs, if short delays can be tolerated.

7. RELATED WORK

Some prior research efforts discussed naming in content-oriented networks and its relationship to security. Notably, [14] proposes establishing bindings between three ICN entities : (1) real-world identity coupled with the the producer of each content object, (2) name, and (3) public key used to verify the object signature. Only two of the three possible bindings (real-world identity–name, name–key and real-world identify–key) are required, while the third can be transitively inherited. However, it is unclear how these bindings can be practically applied in the specific NDN settings.

Self-certifying naming schemes are discussed in [14, 15, 10, 19]. Names are of the form $P : L$ where P is the digest (hash) of the producer's public key, and L is a label set by the producer. It is the latter's responsibility to make sure that names of this form are unique. This guarantees the name–key binding and trades off human readability of names for strong security properties. Although, NDN use human-readable names, name-key binding is achievable by adding the PPKD field to interest messages. This allows interest forwarding based on longest-prefix matching on names. Whereas, using the $P : L$ scheme in NDN would result in tremendously large routing tables. We again recall that self-certifying names in NDN [17] are composed by adding the hash of the content as a name suffix (last component).

Prior work on Denial of Service (DoS) attacks on NDN includes [7] and [2]. Both results addressed a specific DoS attack type – Interest Flooding – based on inundating routers with spurious interest messages. Content poisoning was identified in [12], which also sketched out some tentative countermeasures. Subsequently, [13] proposed the first concrete (however, only probabilistic) countermeasure based on analyzing exclusion patterns for cached content.

Trust and trust management systems are well studied in the literature, especially, in distributed environments, such as MANETs, ad hoc and wireless sensor networks (WSNs). [6] surveys the state of the art in trust management systems for MANETs. It emphasizes the need to combine the notions of "social trust" with "quality-of-service (QoS) trust". A similar survey can be found in [25]. [21] presents an extensive review of trust management systems in WSNs. Based on unique features of WSNs, trust management system's best practices are derived and state of the art countermeasures are evaluated against them. [30] discusses security challenges in designing WSNs. It distinguishes between the definitions of trust and security, and shows that cryptography is not always the solution for trust management. Instead, techniques from other domains should be included in defining and formalizing trust.

Since a single trust metric might not suffice to express trustworthiness of nodes, a multi-dimensional trust management framework is suggested in [20]. Three metrics are used: (1) node collaboration to perform tasks, such as packet forwarding, (2) node behavior, e.g., flagging nodes that flood the network, and (3) correctness of node-disseminated information, e.g., routing updates.

[8] proposes a framework for calculating a network entity's reputation score based on previous interactions feedback. In this framework, each service can apply its own reputation scoring functions. It also supports caching of trust evaluation to reduce network overhead, and provides an API for reporting feedback and calculating reputation scores.

Policymaker [4] is a tool that provides privacy and authenticity for network services. It offers a flexible and unified language for expressing policies and relationships. It also includes a local (per site or network) engine for carrying all trust operations, such as granting access to services.

All aforementioned techniques involve keeping track of other nodes' behavior in order to decide whether they are trusted. However, this general strategy is a poor match for NDN, since routers need an efficient mechanism to trust content, and not other entities. Because content can be served from anywhere it is impractical for routers to trust other entities.

8. CONCLUSION

As argued in this paper, the NDN architecture is inherently susceptible to content poisoning attacks. To mitigate these attacks, we postulated some intuitive trust management goals needed to support content validation in NDN routers. We then presented simple rules that allow all NDN entities to validate content. These rules are compatible with the tenets of the NDN architecture. We also suggested several optimization techniques.

Acknowledgments

This work was funded by NSF under the "CNS-1040802: FIA: Collaborative Research: Named Data Networking (NDN)" project. The authors thank Paolo Gasti, Christopher Wood, CCR reviewers and several anonymous commenters for their feedback on earlier drafts of this paper.

9. REFERENCES

[1] The average web page has almost doubled in size since 2010. http://www.webperformancetoday.com/2013/06/05/web-page-growth-2010-2013/. Accessed: 2014-01-31.

[2] A. Afanasyev, P. Mahadevan, I. Moiseenko, E. Uzun, and L. Zhang. Interest flooding attack and countermeasures in named data networking. In *Proceedings of the IFIP Networking Conference*, 2013.

[3] M. Baugher, B. Davie, A. Narayanan, and D. Oran. Self-verifying names for read-only named data. In *Computer Communications Workshops (INFOCOM WKSHPS), 2012 IEEE Conference on*, pages 274–279. IEEE, 2012.

[4] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proceddings of the IEEE Symposium on Security and Privacy*, pages 164–173, 1996.

[5] Content Centric Networking project (CCNx). http://www.ccnx.org.

[6] J.-H. Cho, A. Swami, and R. Chen. A survey on trust management for mobile ad hoc networks. *IEEE Communications Surveys & Tutorials*, 13(4):562–583, 2011.

[7] A. Compagno, M. Conti, P. Gasti, and G. Tsudik. Poseidon: Mitigating interest flooding DDoS attacks in named data networking. In *Proceedings of the 38th IEEE Conference on Local Computer Networks (LCN)*, 2013.

[8] W. Conner, A. Iyengar, T. Mikalsen, I. Rouvellou, and K. Nahrstedt. A trust management framework for service-oriented environments. In *Proceedings of the 18th International Conference on World Wide Web (WWW)*, pages 891–900, 2009.

[9] M. Conti, P. Gasti, and M. Teoli. A lightweight mechanism for detection of cache pollution attacks in named data networking. *Computer Networks*, 57(16):3178–3191, 2013.

[10] S. K. Fayazbakhsh, Y. Lin, A. Tootoonchian, A. Ghodsi, T. Koponen, B. Maggs, K. Ng, V. Sekar, and S. Shenker. Less pain, most of the gain: incrementally deployable icn. In *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, pages 147–158. ACM, 2013.

[11] K. Fu, M. F. Kaashoek, and D. Mazieres. Fast and secure distributed read-only file system. In *Proceedings of the 4th conference on Symposium on Operating System Design & Implementation-Volume 4*, pages 13–13. USENIX Association, 2000.

[12] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang. DoS DDoS in named-data networking. In *Proceedings of the International Conference on Computer Communications and Networks (ICCCN)*, 2013.

[13] C. Ghali, G. Tsudik, and E. Uzun. Needle in a haystack mitigating content poisoning in named-data networking. In *Proceedings of NDSS Workshop on Security of Emerging Networking Technologies (SENT)*, 2014.

[14] A. Ghodsi, T. Koponen, J. Rajahalme, P. Sarolahti, and S. Shenker. Naming in content-oriented architectures. In *Proceedings of the ACM SIGCOMM workshop on Information-centric networking*, pages 1–6. ACM, 2011.

[15] A. Ghodsi, S. Shenker, T. Koponen, A. Singla, B. Raghavan, and J. Wilcox. Information-centric networking: seeing the forest for the trees. In *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*, page 1. ACM, 2011.

[16] Sandvine global internet phenomena report. `https://www.sandvine.com/trends/global-internet-phenomena/`.

[17] V. Jacobson et al. Named Data Networking (NDN) Project 2012 - 2013 Annual Report. `http://named-data.net/wp-content/uploads/2013/10/ndn-annualreport2012-2013.pdf`.

[18] V. Jacobson et al. Networking named content. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies (CoNEXT)*, pages 1–12, 2009.

[19] T. Koponen, M. Chawla, B.-G. Chun, A. Ermolinskiy, K. H. Kim, S. Shenker, and I. Stoica. A data-oriented (and beyond) network architecture. *ACM SIGCOMM Computer Communication Review*, 37(4):181–192, 2007.

[20] W. Li, A. Joshi, and T. Finin. Coping with node misbehaviors in ad hoc networks: A multi-dimensional trust management approach. In *Proceedings of the 11th International Conference on Mobile Data Management (MDM)*, pages 85–94, 2010.

[21] J. Lopez, R. Roman, I. Agudo, and C. Fernandez-Gago. Trust management systems for wireless sensor networks: Best practices. *Computer Communications*, 33(9):1086–1093, 2010.

[22] D. Mazieres, M. Kaminsky, M. F. Kaashoek, and E. Witchel. Separating key management from file system security. *ACM SIGOPS Operating Systems Review*, 33(5):124–139, 1999.

[23] R. C. Merkle. Method of providing digital signatures, Jan. 5 1982. US Patent 4,309,569.

[24] Named Data Networking project (NDN). `http://named-data.org`.

[25] M. Omar, Y. Challal, and A. Bouabdallah. Certification-based trust models in mobile ad hoc networks: A survey and taxonomy. *Journal of Network and Computer Applications*, 35(1):268–286, 2012.

[26] S. Ramachandran. Web metrics: Size and number of resources. *Online a t https://developers. google. com/speed/articles/web-metrics*, 2010.

[27] D. Smetters and V. Jacobson. Securing network content. Technical report, Technical report, PARC, 2009.

[28] B. Wilcox-OHearn. Names: Decentralized, secure, human-meaningful: Choose two, 2003.

[29] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang. A case for stateful forwarding plane. *Computer Communications*, 2013.

[30] T. Zahariadis, H. C. Leligou, P. Trakadas, and S. Voliotis. Trust management in wireless sensor networks. *European Transactions on Telecommunications*, 21(4):386–395, 2010.