Mitigating On-Path Adversaries in Content-Centric Networks

Cesar Ghali, Gene Tsudik, Christopher A. Wood
Computer Science Department
University of California Irvine
Irvine, CA 92697
Email: {*cghali, gene.tsudik, woodc1*}@*uci.edu*

***Abstract*—Content-Centric Networking (CCN) is a recently proposed Internet paradigm that focuses on scalable, secure and efficient content distribution. The main abstraction is named and addressable content. A consumer requests desired named content by generating a so-called *interest*, which is then routed by the network towards an in-network cached copy, or the authoritative producer, of that content. Since all CCN content must be signed by its producer, consumers and routers can cryptographically verify its correctness, authenticity, and integrity. Thus, in principle, attacks that introduce fake (poisoned) content can be detected. However, verifying content signatures is optional for CCN routers, detection of fake content only implies presence of a malicious upstream entity. A major outstanding problem in CCN is how to react to such attacks, determine their source(s), and re-route interests accordingly.**

In this work, we construct a technique based on efficient per-hop packet integrity checks. Routers share secrets with neighboring routers and use them to verify and generate efficient per-hop packet authenticators. An on-path attacker that tampers with content in transit is quickly detected by downstream routers. Moreover, an on-path attacker that hijacks a namespace is discoverable. Our experimental assessment indicates that the proposed technique incurs very low per-packet overhead. Furthermore, since our approach is not CCN-specific, it can be applied to IP-based networks as well.

***Index Terms*—Content-Centric Networking, on-path attackers, packet integrity, adversary leap frog**

I. Introduction

Opportunistic in-network caching is a core feature of Information-Centric Networking (ICN) architectures, such as Content-Centric Networking (CCN) [1] and Named-Data Networking (NDN) [2]. In these architectures, routers are (optionally) equipped with caches that temporarily store recently forwarded content objects. These cached copies can satisfy future requests (called "interests") for the same content, thus reducing end-to-end latency and decreasing overall utilized network bandwidth.

Each content is cryptographically bound to its name, in most cases, via a digital signature generated by that content's producer. Consumers must, and routers can, verify these signatures to ensure they receive the correct content. Routers are only required to verify content *before serving it from a cache* [1]. This is done to prevent so-called *content poisoning attacks*, wherein the adversary Adv injects fake content, i.e., content with an invalid signature, into the network. Content poisoning is not a new threat; it was first identified in [3]. Subsequently, Ghali et. al proposed a means to mitigate it in [4] and [5]. One comprehensive counter-measure is the Interest-Key Binding (IKB) rule, which reduces the problem of content poisoning to that of network-layer trust [5]. IKB requires consumers and producers to collaborate in order to provide routers with enough trust information about requested content to allow them to detect poisoned content.

However, even with today's networking technology, performing a single signature verification per content is not aways practical at line speed [5]. Moreover, not every router would cache and verify each content. In the presence of an on-path attacker that hijacks a prefix or simply modifies content in transit, poisoned content would not be discovered until it gets close to the consumer.

The only way around an on-path attacker is to bypass it by adjusting routers' forwarding information. However, without knowledge of the attack origin or network topology, such adjustments might be erroneous or suboptimal due to being done very far away from the actual adversary. To aid routers in making informed forwarding adjustments, DiBenedetto et al. [6] proposed an adaptive forwarding strategy based on consumer-provided content "complaints." Specifically, a consumer that detects poisoned content informs upstream routers about said content and provides them with information needed to check its validity. A router verifies such downstream complaints and adjusts its forwarding strategy to avoid the path that contains the attacker. These complaints are recursively forwarded upstream so that every router can adjust its forwarding table. Despite its seeming simplicity, this approach has several undesirable features:

- Complaint messages are a form of computational DoS on routers, due to cryptographic operations needed to validate them. If complaint messages are not protected, another type of DoS would occur since the attacker could generate spurious complaints.
- Complaint messages can be very large, as they encompass referenced fake content and all information needed to authenticate it. This translates into communication DoS on routers.
- Routers are forced to update routes even though immediate upstream routers (which provided content) may not be malicious. This can lead to suboptimal route selection that harms downstream routers. For example, a router could unnecessarily switch to a more expensive link.

More importantly, [6] proposes an application-layer remedy to a network-layer problem. Clearly, on-path attackers *are not unique to ICN*. They may occur in IP networks under the guise of transit routers that modify transit packets' payloads, or drop packets altogether. Therefore, we believe that on-path attackers must be dealt with at the network layer. In other words, while content authenticity is strictly an end-to-end issue, content retrieval, i.e., via avoidance of on-path attackers, is a network-layer concern.

In this paper we reconcile content authenticity and retrieval with an efficient per-hop content integrity check. The main idea is to allow routers to detect when upstream peers have modified a content packet. Intended contributions of this work are:

1) A novel "adversary leap frog" scheme that uses cryptographic MACs to ascertain authenticity of upstream content packets.
2) A merge of IKB with the above scheme to enable in-network content authenticity and secure content retrieval.
3) A security analysis and experimental assessment of the proposed scheme to show its utility and practicality.

The rest of the paper is organized as follows: Section II presents an overview of CCN. Next, Section III describes content poisoning and the difficulty of its mitigation. The on-path attacker threat model is described in Section IV. Our approach is then detailed in Section IV. Then, security analysis and experimental assessment are presented in Sections VI and VII, respectively. The paper concludes with a summary of related work and future directions, in Sections IX and X.

II. CCN Overview

CCN [1, 7] is a network architecture based on named content. Rather than addressing content by its location, CCN refers to it by a human-readable name, which is composed of one or more variable-length components that are opaque to the network. Component boundaries are explicitly delimited by "/" in the usual UNIX path representation. For example, the name of a BBC news content for April 1, 2017 might look like: `/bbc/news/2017april01`. Since CCN's main abstraction is content, there is no means to directly address "nodes" (interfaces, hosts and routers), although their existence is assumed.

In CCN, content is only delivered to consumers upon explicit request, called an *interest*. A CCN router never forwards a content unless it is preceded by an interest requesting that content. If a content is received without prior interest, it is considered unsolicited and is dropped. Consumers request content with name N by sending an interest with the same name. A content named N' is considered a match for – and therefore can be sent in response to – an interest for N if and only if $N = N'$. In other words, content name matching in CCN is exact.

There are three types of CCN entities[1]: (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 two components [1]:

- *Forwarding Interest Base* (FIB) – routing table of name prefixes and corresponding outgoing interfaces (to route interests).
- *Pending Interest Table* (PIT) – table of currently not-yet-satisfied (pending) interests and a set of corresponding incoming interfaces.

An entity may also have a *Content Store* (CS) – a cache used to temporarily store content. (From here on, we use the terms *CS* and *cache* interchangeably). Unlike the PIT and FIB, a cache is optional.

When a router receives an interest for name N, and there are no pending interests for a matching 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 already an entry for the same content name in the PIT, the router collapses the present interest (and any subsequent ones for N) and stores only the interface on which it was received. When content is returned, the router forwards it out on all interfaces on which an interest for N has arrived and flushes the corresponding PIT entry. Since no additional information is needed to deliver content, an interest does not carry a "source address".

A CCN content packet includes several fields. Beyond the name field, we are only concerned with the `Validation` field. This contains the content verification algorithm details and a public key signature generated by the content producer. The latter covers the entire content, including all explicit components of the name and a reference to the public key needed to verify it.

A CCN interest packet has the following fields:

- `Name` – CCN name of requested content.
- `ContentObjectHashRestriction` (ContentId) – cryptographic hash digest of requested content.
- `KeyIdRestriction` (KeyId)– cryptographic hash digest of the public key used to verify requested content.

If an interest specifies a KeyId then it must match the cryptographic hash digest of the public key in the corresponding content object. If an interest specifies a ContentId then it must match the cryptographic hash digest of the entire content object (modulo packet headers). These are considered restrictions, since they limit the scope of acceptable content object responses to an interest. Finally, we note that an interest which carries a ContentId is said to have a "self-certifying name."

III. Content Poisoning and Namespace Arbitration

Content poisoning is an attack on *content retrieval* that prevents consumers from obtaining valid (authentic) data. This attack comes in two flavors: proactive and reactive, as defined in [3]. In this paper, we focus on the latter and use the term *content poisoning* stricly in that context.

Reactive content poisoning attacks occur when an on-path Adv injects fake content objects into the network. A fake con-

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

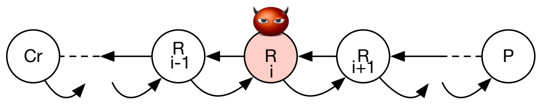

Fig. 1. An on-path content poisoning attack.

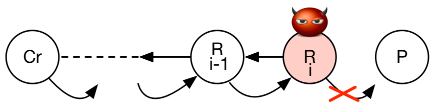

Fig. 2. An on-path content generation attack.

tent object is one that has an invalid or malformed signature, or a valid signature generated with the wrong key. Consumers can easily identify fake content by verifying content signatures. Not all routers, however, can easily do so, since verifying content signatures at line speed can be too computationally expensive. Thus, since not all routers verify content signatures, Adv can inject content that percolates through the network until some entity verifies its. When a downstream router or a consumer detects fake content, it only learns that *some upstream router or a producer tampered with that content*. The verifying entity has no insight into *where* the attack occurred.

Consider a path of legth n with intermediate routers R_{i-1}, R_i, and R_{i+1}, between a consumer Cr and producer P shown in Figure 1. Downstream arrows indicate the flow of content from P to Cr. Adv is an on-path attacker that modifies this content in transit. Upon authenticating content, any router R_j, $j \in [1, i-1]$, only learns that either some router $R_k, k \in [j+1, n]$ or P provided the fake content. Adv can attack in two ways: (1) by modifying content packets in transit, or (2) by responding with fake content. We refer to them, respectively, as *on-path modification* and *on-path generation* attacks. They are shown in Figures 1 and 2. Detecting on-path attacks by an intermediate router is equivalent to checking content integrity and authenticity as content flows through the network.

To obviate the need for signature verification in the fast path, [6] proposed a reactive technique to correct network behavior when attacks are detected. This proposal is premised on the argument that undetected modification and generation attacks result from producers publishing under unauthorized namespaces. This is addressed by requiring the consumer to report fake content to routers that verify these reports and forward them upstream, if necessary. Routers then greedily find an alternate path to the valid content, by either: choosing a different forwarding interface for the content, or probing all interfaces until valid content is returned. Although this mitigation strategy works in theory, it has certain drawbacks discussed in Section I.

Moreover, the approach of [6] is based on the dubious assumption that namespace ownership is proven by content published under that namespace. This means that any application can publish under any namespace provided that it possess the public key that a consumer trusts. However, there is nothing stopping multiple applications from claiming ownership of the same namespace. To see why this is problematic, consider two applications: A_1 and A_2 that both claim ownership over namespace $N = /\mathtt{a}/\mathtt{b}$, i.e., both advertise themselves as producers under N. Moreover, neither one is malicious. Since each application has its own key-pair, data produced by A_1 and A_2 is signed and then verified with *different* public keys. Now assume that consumer Cr wishes to obtain content: $/a/b/x$ from A_1 using KeyID K_1. Cr issues an interest and provides K_1's ID. However, if A_2 is closer to Cr than A_1, A_2 would produce content signed by K_2, not K_1. Therefore, all on-path routers as well as Cr would deem the data invalid. However, whether it is invalid depends on who has proper ownership over N. Clearly, this problem occurs since the network allows both applications to advertise under N without verifying ownership.

Without a namespace arbitration mechanism, this problem is unavoidable. Thus, we conclude that a node can not just unilaterally advertise under *any* namespace it wants. There must exist some globally trusted authority that manages namespace ownership. This authority can help prove ownership of a namespace. This way, namespace ownership can be ascertained before routes are injected into the network. As a result, any after-the-fact content modification or generation attack would stem from failure to check packet authenticity and integrity in the fast path. Addressing this problem would thus mitigate on-path attackers.

IV. Threat Model

We now outline our system and threat model. We assume a network composed of routers, producers, and consumers. Routers forward interest and content packets between consumers and producers. Routers have identities (for network management purposes) and can learn identities of nearby peers in the network. Also, routers may be organized into autonomous systems (ASes). Routers are added to the network in a controlled and secure fashion.

Each namespace is owned by an application (or identity) and is served by at least one producer on behalf of the application. Each namespace is also associated with a unique public-private key-pair. It is possible for multiple producers to serve the same namespace. In this case, they would all belong to the same application. Lastly, there exists a globally trusted authority responsible for managing namespace ownership. It may be implemented as a centralized arbiter similar to IANA, a decentralized system such as DNS, or a distributed system such as Namecoin [8]. For simplicity, we assume that the authority is a key-value store which maps namespaces to the corresponding public key or an empty string (if no application owns the namespace). Moreover, we assume that the authority is queried before producers are allowed to inject namespace prefixes into the routing protocol.

As mentioned earlier, we assume a general on-path adversary Adv that can compromise a polynomial number of routers (up to k) and any producer. Adv has complete control over each compromised entity. In particular, it controls the content and timing of messages generated and processed by such entities. Adv can also spawn authenticated nodes under its control on demand. This capability can be used to modify packets or generate fake content. Security against Adv means that it is infeasible for Adv to perform these actions without detection. Once Adv is detected, honest entities can take action to bypass it.

V. Integrity Zones

As mentioned above, on-path attacks are possible because there is no efficient means for routers to verify packet integrity and origin authenticity on the fast path. Meanwhile, application-driven solutions (e.g., [6]) have some prominent drawbacks. We believe that this problem should be addressed at the network-layer. With respect to integrity, hop-by-hop packet mechanisms are insufficient since Adv can compromise individual routers. At the same time, end-to-end integrity mechanisms are infeasible since "ends" are undefined: one end could be the producer or an intermediate router's cache. Ideally, we need something between these two extremes. For origin authenticity, digital signatures are insufficient for identifying the origin of fake content. A router should be able to learn if a packet was generated by a producer who owns the corresponding namespace. This is *not* the same as verifying that the packet is authentic, since the verifying public key need not be the same as the namespace ownership key.

Our approach relies on *integrity zones*, which work as follows: Every router shares (and rotates) a unique key with every router that is $k \geq 2$ hops away. These keys are used to generate and verify packet MACs to ensure packet integrity in transit. Upon receipt of a content packet, a router verifies at most k MAC tags and, if all are valid, replaces them with k new MAC tags generated locally. Upon receipt of an interest packet, a router simply appends its ID to the packet. If the number of downstream IDs exceeds k, it also drops the oldest one. In effect, the MAC tags and router IDs form a sliding window of size k that moves as packets propagate through the network. If Adv compromises at least k contiguous routers, the system fails.

Integrity zones require knowledge of the distance to the content producer, which can be provided by a secure distance-vector routing protocol, such as DCR [9]. This is needed so that routers can check if content was really generated by the appropriate producer. When generating content in response to an interest (which carries identities of the k previous hops through which the packet flows) the producer computes and includes k MACs using the keys associated with these k routers. Each downstream router then validates the producer-generated MAC before including its own MAC, as described above. Also, the i-th downstream router, where $i \leq k$, *must* check that there are at least i valid MACs in the content packet. This ensures that the original MACs were not generated by an on-path Adv that hijacked a namespace.

Integrity zones allow routers to play "adversary leap frog." In other words, when less than k contiguous nodes are compromised, a router can always learn exactly which router is malicious and can take steps to remedy the situation in the forwarding plane. Inspired by [10], the crux of the proposed technique is for routers to verify each content packet as it is

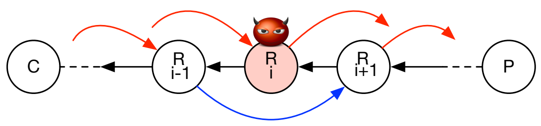

Fig. 3. Adv controlling intermediate router. Blue arrow shows dependency between R_{i-1} that detects a problem and R_{i+1} that does not. $k = 2$.

routed through the network. Consider the following example, where:

$$\mathsf{path} = R_1, R_2, R_3, \ldots, R_{n-2}, R_{n-1}, R_n$$

is a path between some Cr and P. Moreover, assume that Cr previously issued an interest for P that resulted in a content C being forwarded along the reverse of path. Now, suppose that Adv controls $R_i \in \mathsf{path}$, and modifies either the content packet or its signature. This means that R_{i+1} sees a valid content packet while R_{i-1} does not. The goal is for R_{i-1} to learn that R_i is misbehaving. Ideally, R_{i+1} computes a content MAC using a key shared with R_{i-1}. Then, R_{i-1} fails to verify this MAC and learns that R_i is malicious. This flow is shown in Figure 3. The network can then attempt to avoid R_i for *all interests* and heal itself without requiring any consumer involvement.

To summarize, within a given integrity zone, each router is responsible for: (a) appending its identifier to each interest, (b) verifying and removing k MACs of each content packet, and (c) computing k MACs and appending them to each content packet. Thus, every router performs $2k$ MAC operations. (See Section VII.) Below, we discuss this approach in detail, including: (1) how those routers obtain shared keys, (2) why producer (anchor) distances are required, (3) packet formats and processing logic, and (4) network recovery steps.

1) Leap-Frog Key Distribution: Each router shares a key with every router $k \geq 2$ hops away. Depending on the network ecosystem, these keys may be pre-shared between routers, i.e., within a single AS. Otherwise, keys must be established using a key exchange protocol. One potential obstacle is that such protocols require some form of a PKI to ascertain each peer's identity. However, this is not a show-stopper, since most networks are managed and contain routers controlled by the same administration. We therefore defer this bootstrapping phase to future work.

2) Anchor Distances: Integrity zones allow a router to prove that its k upstream routers did not modify a content packet. However, if Adv hijacks a namespace there could be $< k$ MACs in a content packet. This is only possible if the consumer-to-producer path has $j < k$ hops. The packet would then carry j MACs and each router on the path would know this distance and expect the appropriate number of MACs.

There is one more edge-case: Let R be a benign router $j \geq 2$ hops away from P. Suppose that Adv is adjacent to R on the $R \to P$ path. We assume that Adv attempts a data generation attack. In doing so, it must provide k valid MACs verifiable by k downstream routers, including R. Upon receipt of the fake content packet, R can only verify *one* MAC – the

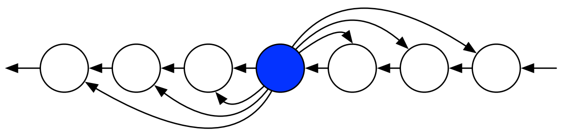

Fig. 4. Upstream MAC dependencies and downstream MAC generation.

Algorithm 1 ProcessInterest(k)

```
Input: $I[N, \mathsf{IDs}]$, $F_{in}$, $R_{\mathsf{ID}}$
if $N$ in the PIT then
    Add $F_{in}$ and IDs to the PIT entry with name $N$.
else
    Append $R_{\mathsf{ID}}$ to IDs to form IDs'
    Lookup $F_{out}$ in the FIB using $N$
    Forward $I[N, \mathsf{IDs}']$ to $F_{out}$
end if
```

one generated by Adv. If R does not know that P is after R on the path, R may incorrectly conclude that the content packet is valid and forward it. However, if R knows that P is j hops away, it knows to verify j MACs. R can therefore immediately learn that an attack took place. Thus, security of this scheme relies on accurate and secure anchor distances. We assume the existence of a secure protocol that allows routers to learn the minimum distance to a namespace anchor. This should be feasible assuming a global namespace ownership authority discussed earlier. However, we defer exploration of this topic to future work.

3) Packet Formats and Processing: Each content packet carries MACs needed to check validity of previous k hops (see Figure 4). Also, each packet carries MACs necessary for the i-th *downstream* router to verify previous $k-i$ hops. Thus, a content packet carries no more than $k(k+1)/2$ MACs at a time. MACs generated by each router are stored in a list. Also, each MAC is associated with a symmetric key shared between two routers. For example, a MAC generated by a key shared by R_i and R_j with IDs i and j is $\sigma_{i,j}$. Each MAC has an associated ID of the router that generated it, denoted $\sigma_{i,j}.ID$. A router stores its shared keys in a table called KeyTable. A key is fetched using the identifier, e.g., $k_{i,j} = \mathsf{KeyTable}[\sigma_{i,j}.ID]$. In S-expression notation, the set of MAC lists stored in a packet header is represented as follows:

```
(T_INTEGRITY_ZONE
                        (T_PACKET_MACS   [$\sigma_{i,i+1}$]
                        (T_PACKET_MACS   [$\sigma_{i,i+1}, \sigma_{i,i+2}$])
                        ...
                        (T_PACKET_MACS   [$\sigma_{i,i+1}, \sigma_{i,i+2}, \dots, \sigma_{i,i+k}$])
                        )
```

When a router processes a content packet it deletes at most k MACs since they are useless to downstream routers. This procedure is detailed in Algorithm 2. (Changes from standard CCN are underlined.) Processing an interest is much easier: a router simply injects its own identifier in the interest header and forwards it, if necessary. This is detailed in Algorithm 1. (Changes from standard CCN are underlined.)

Currently, all MACs must fit in the per-hop header of the content packet. However, the current CCN packet format restricts this header to a maximum size of 256 bytes. Assuming a MAC of 128 bits, or 16 bytes, each packet can support a radius of at most $k = 4$.[2] This a reasonable restriction, since it requires Adv to compromise $k \geq 4$ contiguous routers to subvert integrity zones.

Algorithm 2 ProcessContent(k)

```
Input: $C[\mathsf{MACs}, D, V]$,
Output: $C[\mathsf{MACs}', D, V]$, $F_{out}$
for $l_i$ in MACs do
    $\sigma := l_i.\mathsf{pop}()$
    $k_{ID} := \mathsf{KeyTable}[\sigma.ID]$
    if $\sigma \neq MAC(k_{ID}, D||V)$ then
        Drop $C$ and flag upstream router as malicious
    end if
end for
Drop first MAC for each $l_i$ in MACs
$Entry := PIT.\mathsf{Lookup}(C)$
if $|\mathsf{MACs}| < Entry.d$ then
    Drop $C$ and flag upstream router as malicious
end if
$l_R = []$
for $ID$ in $Entry.IDs$ do
    $k_{ID} := \mathsf{KeyTable}[ID]$
    $l_R.\mathsf{Append}(MAC(k_{ID}, D||V)$
end for
$C' := C.\mathsf{MACs}.\mathsf{Append}(l_R)$
return $C'$, $Entry.F_{out}$
```

4) MAC Compression: Packet format described in the previous section requires all MACs to be listed separately in each content packet. The resulting overhead is $\mathcal{O}(k^2)$ MACs per packet. To lower it, MACs can be compressed. Consider the verification process at each hop. Each router verifies k MACs individually and detects an attack if at most one fails. These k MACs could be aggregated into a single MAC. Each router could then verify an aggregate. With this format, the packet header would only need to carry a list of k aggregate MACs and list of k upstream router IDs, as shown below:

```
(T_INTEGRITY_ZONE
                        (T_ROUTER_IDS   [$\mathtt{ID}_{i+1}, \mathtt{ID}_{i+2}, \dots, \mathtt{ID}_{i+k}$]
                        (T_PACKET_MACS   [$\sigma_{i+1}, \sigma_{i+2}, \dots, \sigma_{i+k}$])
                        )
```

With this format, each packet is processed as shown in Algorithm 3. This variant also modifies per-hop headers in place, which is far more efficient.

5) Recovery: When a R_i detects that *only one* of its upstream neighbors $R_j, j > i$ tampered with a content packet, it concludes that R_{i+1} is also malicious. This is because, if R_{i+1} were not malicious, it would have detected the attack by R_j, $j > i+1$, and dropped the packet accordingly. Therefore, R_i can remove all FIB entries that point to R_{i+1}. It does not need to probe the network to find a next-best route. If there are no longer any viable routes, R_i must generate interest NACKs.

[2]This limit assumes there are no other per-hop headers, such as the interest lifetime or recommended cache time. In practice, these headers are usually present in packets, meaning that $k < 4$. However, given that the packet format is flexible, we could easily extend the per-hop header capacity to accommodate these new fields.

Algorithm 3 ProcessContentWithCompressedMACs(k)

```
1: Input: $C[\mathsf{IDs}, \mathsf{MACs}, D, V]$,
2: Output: $C[\mathsf{IDs}', \mathsf{MACs}', D, V]$, $F_{out}$
3: $\sigma = 0^\lambda$ {Empty string to start}
4: for $i := 1, \ldots, k$ do
5:     $k_{ID} := \mathsf{KeyTable}[\mathsf{IDs}[i]]$
6:     $t = MAC(k_{ID}, D||V)$
7:     $\sigma = \sigma \oplus t$ {Aggregate the MAC}
8: end for
9: if $\sigma \neq \mathsf{MACs}[1]$ then
10:     Drop $C$ and flag upstream router as malicious
11: end if
12: Drop the first element of MACs and shift to the left
13: $Entry := PIT.\mathsf{Lookup}(C)$
14: if $|\mathsf{MACs}| < Entry.d$ then
15:     Drop $C$ and flag upstream router as malicious
16: end if
17: for $i := 1, \ldots, |Entry.IDs|$ do
18:     $k_{ID} := \mathsf{KeyTable}[Entry.IDs[i]]$
19:     $\mathsf{MACs}[i] = \mathsf{MACs}[i] \oplus (MAC(k_{ID}, D||V)$
20: end for
21: return $C$, $Entry.F_{out}$
```

VI. Security Analysis

Security of our integrity zones scheme is premised on whether Adv can compromise more than k contiguous routers. If $k = 2$ and Adv compromises two contiguous routers, any part of the packet can be modified without any downstream router detecting the problem. However, if k is large enough such that Adv can not succeed, then integrity zones protects against generation and modification attacks. We argue this below.

Claim 1: Assuming a MAC scheme secure against existential forgeries, and Adv that is unable to: (a) compromise k contiguous routers along paths of length at least k routers, or (b) compromise l contiguous routers along paths of length $l < k$, and a distance-vector routing protocol that provides correct anchor distances, the integrity zones scheme is secure against content modification and generation attacks.

Proof: (Sketch) First, we show that modification attacks are impossible. This follows from security properties of the underlying MAC scheme. If $c < k$ contiguous routers are compromised and a packet is modified, then at least one MAC tag in a downstream router would fail verification. This MAC corresponds to the upstream router which is malicious.

Assuming a namespace ownership authority, a producer can only advertise a namespace which it owns. Therefore, data generation occurs when: (a) an on-path router maliciously intercepts interests and responds with fake content, or (b) the producer provides fake content. We address these cases together since they differ only in the distance to the producer. Assume that the distance to the producer for the fake content from the verifying router is d. Also, suppose that a packet with l MACs verifies correctly. There are three cases with respect to l, k, and d: (1) $l = k$ and $d \geq k$, (2) $l = k$ and $d < k$, and (3) $l < k$. (Note that $l > k$ is not possible based on the packet format.) In (1) and (2), R fails to detect the attack only if a MAC was forged.[3] Whereas, (3) only occurs if the packet traverses a path shorter than k hops. Per our assumption, Adv can not compromise routers along this path and thus an attack can not occur.

[3]A MAC forgery occurs when Adv generates a valid MAC without knowledge of the secret key.

We note that interests only carry router IDs and not MACs. Tampering with this list by appending, removing, or changing IDs would only cause downstream routers to fail verification. This is because upstream routers would use wrong keys to generate MACs. ■

As indicated above, one limitation of the integrity zone scheme is with paths of length $l < k$ routers. If Adv can compromise all l routers, the scheme fails to detect the attack.

VII. Experimental Analysis

We now assess the overhead of the integrity zones scheme. From a performance perspective, overhead is incurred at every hop. Specifically, the scheme involves at most $2k$ MAC operations: k verifications and k generations. Therefore, network topology has no impact on per-packet overhead. Thus, our assessment has two parts: (1) measuring efficiency of various MAC schemes, and (2) measuring impact of integrity zones on consumer latency.

A. MAC Overhead

Let m be a message of $|m|$ blocks that serves as MAC input and K be the corresponding key. There are many popular MAC schemes, e.g.,: CMAC [11] and HMAC [12]. HMAC is defined as follows:

$$HMAC(K, m) = H((K' \oplus \mathsf{opad})||H((K' \oplus \mathsf{ipad})||m)),$$

where opad and ipad are constants defined in the standard and K' is a key derived from the master key K. CMAC, or CBC-MAC, is defined (roughly) as follows:

$$\begin{aligned} B_0 &= AES(K, m_0) \\ B_i &= AES(K, B_{i-1} \oplus m_i) \\ B_{|m|} &= AES(K, B_{i-1} \oplus m_i \oplus K') \\ CMAC(K, m) &= B_{|m|} \end{aligned}$$

where K' is a key derived from K and B_i is the i-th output of encrypting m in CBC mode.

Both HMAC and CMAC require a complete pass over m. The choice depends on what is available on the run-time platform. In general, we view HMAC and CMAC as approximately equivalent in terms of performance (in the absence of AES acceleration). Figure 5 shows the performance of HMAC as a function of message size. We see that overhead reaches the millisecond mark when k approaches 6 and $|m|$ exceeds 4KB.

However, this can be improved with the following optimization. Notice that both HMAC and CMAC process all of m to produce a digest. Cryptographically, these functions provide the equivalent security guarantees if a cryptographic hash digest of m, i.e., $H(m)$ for some pre-image resistant hash function, is the input to MAC. Pre-image resistance stipulates that it must be infeasible to find another message m' such that $H(m) = H(m')$, i.e., forgery remains infeasible. With this in mind, our optimization works by first computing $H(m)$ and

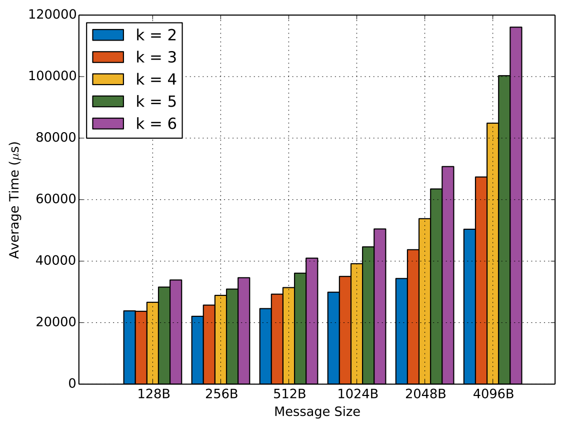

Fig. 5. HMAC overhead as a function of message size and k.

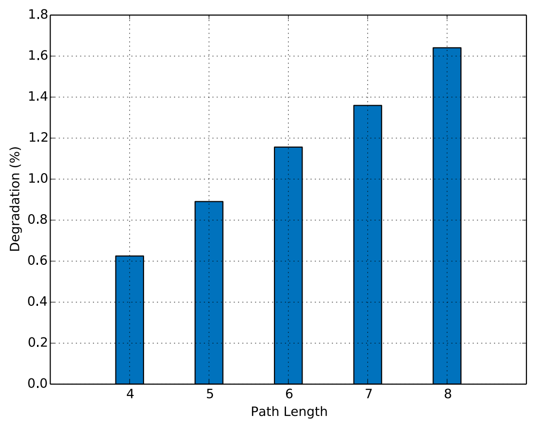

Fig. 7. Perceived latency reduction due to per-packet integrity zones checks

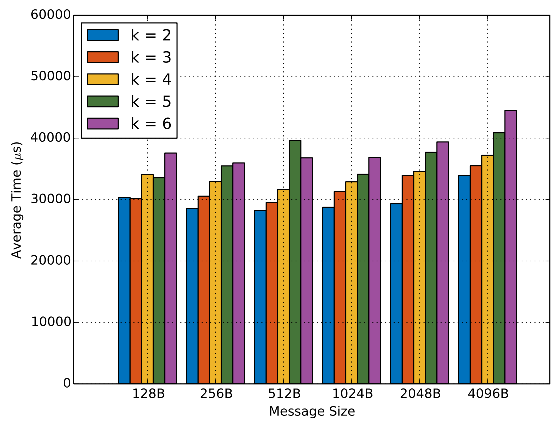

Fig. 6. Hashed HMAC processing time as a function of the message size and k.

using that as input to each MAC operation. Figure 6 shows this optimization as a function of message size and k. (We use SHA-256 as $H(\cdot)$, given its widespread use in CCN.) Results indicate a significant reduction in processing time for nearly all data points.

B. Network Impact

To assess impact on consumer latency, we modified the ccns3Sim simulator [13, 14], based on ns3 [15], to support our scheme. We then created a simple N-node path between Cr and P. Every second, Cr issues an interest for a random content produced by P. We measure end-to-end latency as a function of N with $k = 2$, which includes per-packet overhead induced by each router. Results are shown in Figure 7, which plots degradation, i.e., percentage latency increase over standard CCN.

VIII. Discussion

A. Scalability

The integrity zones scheme entails lightweight hop-by-hop packet integrity checking that is both computation- and bandwidth-efficient. However, it still incurs certain overhead. Since each router must share a key with every router $i = 2, \ldots, k$ hops away, the number of keys can become quite large. For example, suppose that R has 5 neighbors, each of which has its own 5 (distinct) neighbors, excluding R. Then, if $k = 2$, R would have to store 25 keys for each of these routers. Thus, storage overhead is a function of k and network topology. In a highly connected network where routers have many neighbors, this may be prohibitive. However, in such a scenario, it is far less likely for Adv to compromise all routers $i = 2, \ldots, k$ hops away. Therefore, k can actually be lower. When there are fewer path choices for a router, it pays to have a larger k, since the Adv's job becomes harder.

The per-packet overhead in each packet could be dealt with by only generating and verifying MACs at AS boundaries. This would require ASes to share keys with other ASes in the network. This may be more plausible than individual routers sharing keys since trust relationships at the AS level are more controlled. Specifically, two AS operators can decide whether or not to trust each other and, if so, allocate a key to the corresponding gateways. Although this may simplify key management, it makes the attack origin detection less accurate. When MACs are verified and generated by individual routers, i.e., not just gateways, a router learns precisely where the on-path attack took place. At the AS level, a router (gateway) only learns that an attack took place within an AS.

B. Global Zone Size

We assumed that k is a global constant. However, this need not be the case. Indeed, k may be AS-specific. It is the responsibility of a gateway between AS-s to bridge the gap between different zones sizes. For example, suppose that an interest is generated in AS D_1 with $k = k_1$. It traverses a gateway G to another AS D_2 with $k = k_2$. G must share keys with all of its neighbors at most $\max\{k_1, k_2\}$ hops away. When it receives the interest, it verifies k_1 MACs (if present) and injects its own. Upon receipt of the corresponding content, it verifies k_2 MACs and injects $\max\{k_1, k_2\}$ MACs before forwarding it downstream. If $k_1 < k_2$, each router in D_1 would be able to verify its k_1 MACs. However, if $k_1 \geq k_2$, no router in D_1 that is k_1 or more hops away from G can verify MACs generated upstream of G.

C. Privacy

At each hop, the integrity zones scheme requires interests (respectively, content objects) to identify k downstream (respectively upstream) routers using their KeyId-s. This reveals path information to each router, something that was not previously visible in CCN. This may be problematic at the network edges, specifically, when a R is less than k hops away from a consumer-facing router.

IX. Related Work

On-path attacks were only directly addressed by DiBenedetto et al. [6]. As previously discussed, this work depends on the adaptive forwarding layer of NDN (and CCN), as discussed in [16]. Wu et al. [17] proposed a similar approached called Router-Oriented Mitigation (ROM). Their scheme generalizes the binary trust value of [6] and instead assigns each router a reputation. This reputation is then taken into account when making forwarding decisions. The goal is to eliminate, or at least bypass, on-path attackers in the network. Approaches to make content verification more efficient were proposed in [18]. The authors rely on intelligent cache management without any changes to the verification mechanism, i.e., the scheme still relies on signature verification.

Previous work [4, 5] focused on the more generic problem of content poisoning, irrespective of its relation to forwarding behavior.

X. Conclusions

We proposed a means of mitigating on-path packet modification and generation attacks. Unlike previous work which: (a) assumes no authoritative entity responsible for controlling namespace ownership, and (b) sub-optimally and slowly routes around on-path attackers, our approach allows routers to immediately, i.e., in constant time, detect on-path attacks. Moreover, control plane modifications made to bypass the attacker are made *closest* to the attacker, rather than closest to the detecting consumer or router. Our approach is based upon the observation that the attacker is very unlikely to compromise several contiguous routers across administrative domains. We show that the proposed scheme can be easily implemented with the current CCN packet format and introduces very little per-packet overhead. This makes our scheme widely applicable to general network topologies and deployments.

XI. Acknowledgment

The authors are grateful to LCN 2017 reviewers for their useful and insightful comments. Christopher A. Wood was supported by the NSF Graduate Research Fellowship DGE-1321846.

References

[1] M. Mosko, I. Solis, and C. Wood, "Ccnx semantics," *IRTF draft-mosko-icnrg-ccnxsemantics-04, ICNRG*, 2017.

[2] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, P. Crowley, C. Papadopoulos, L. Wang, B. Zhang *et al.*, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[3] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang, "Dos and ddos in named data networking," in *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*, July 2013, pp. 1–7.

[4] C. Ghali, G. Tsudik, and E. Uzun, "Needle in a haystack: Mitigating content poisoning in named-data networking," in *Proceedings of SENT Workshop*, 2014.

[5] C. Ghali, G. Tsudik, and E. Uzun, "Network-layer trust in named-data networking," *ACM CCR*, vol. 44, no. 5, pp. 12–19, 2014.

[6] S. DiBenedetto and C. Papadopoulos, "Mitigating poisoned content with forwarding strategy," in *Computer Communications Workshops (INFOCOM WKSHPS), 2016 IEEE Conference on*. IEEE, 2016, pp. 164–169.

[7] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard, "Networking named content," in *Co-NEXT*, 2009.

[8] H. Kalodner, M. Carlsten, P. Ellenbogen, J. Bonneau, and A. Narayanan, "An empirical study of namecoin and lessons for decentralized namespace design," in *Workshop on the Economics of Information Security (WEIS)*. Citeseer, 2015.

[9] J. J. Garcia-Luna-Aceves, "Name-based content routing in information centric networks using distance information," in *Proceedings of the 1st international conference on Information-centric networking*. ACM, 2014, pp. 7–16.

[10] M. T. Goodrich, "Leap-frog packet linking and diverse key distributions for improved integrity in network broadcasts," in *Security and Privacy, 2005 IEEE Symposium on*. IEEE, 2005, pp. 196–207.

[11] J. Song, R. Poovendran, J. Lee, and T. Iwata, "The aes-cmac algorithm," Tech. Rep., 2006.

[12] H. Krawczyk, R. Canetti, and M. Bellare, "Hmac: Keyed-hashing for message authentication," 1997.

[13] "CCNx Module for NS3," https://github.com/chris-wood/ccns3Sim, accessed: May 14, 2016.

[14] "CCNx on-path simulator," https://github.com/chris-wood/ccn-onpath-simulation-ccnsim, accessed: May 14, 2016.

[15] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," *Modeling and tools for network simulation*, pp. 15–34, 2010.

[16] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane," *Computer Communications*, vol. 36, no. 7, pp. 779–791, 2013.

[17] D. Wu, Z. Xu, B. Chen, and Y. Zhang, "What if routers are malicious? mitigating content poisoning attack in ndn," in *Trustcom/BigDataSE/ISPA, 2016 IEEE*. IEEE, 2016, pp. 481–488.

[18] D. Kim, S. Nam, J. Bi, and I. Yeom, "Efficient content verification in named data networking," in *Proceedings of the 2nd International Conference on Information-Centric Networking*. ACM, 2015, pp. 109–116.