Capability-Based Security Enforcement in Named Data Networking

Qi Li, *Senior Member, IEEE*, Patrick P. C. Lee, *Senior Member, IEEE*, Peng Zhang, *Member, IEEE*, Purui Su, *Member, IEEE*, Liang He, and Kui Ren, *Fellow, IEEE*

***Abstract*— Named data networking (NDN) enhances traditional IP networking by supporting in-network content caching for better bandwidth usage and location-independent data accesses for multi-path forwarding. However, NDN also brings new security challenges. For example, an adversary can arbitrarily inject packets to NDN to poison content cache, or access content packets without any restrictions. We propose capability-based security enforcement architecture (CSEA), a capability-based security enforcement architecture that enables data authenticity in NDN in a distributed manner. CSEA leverages capabilities to specify the access rights of forwarded packets. It allows NDN routers to verify the authenticity of forwarded packets, and throttles flooding-based DoS attacks from unsolicited packets. We further develop a lightweight one-time signature scheme for CSEA to ensure the timeliness of packets and support efficient verification. We prototype CSEA on the open-source CCNx platform, and evaluate CSEA via testbed and Planetlab experiments. Our experimental results show that CSEA only incurs around 4% of additional delays in retrieving data packets.**

***Index Terms*— Security, capability, named data networking.**

I. Introduction

NAMED Data Networking (NDN) has been proposed to replace the "connection-based" model in traditional IP networking with the "content-based" model. By identifying data packets by names instead of locations, NDN enables flexible in-network caching for improved bandwidth usage and allows location-independent content access for multi-path forwarding [14]. From a security perspective, the name-based transmission model of NDN does not reveal who requests data packets and who hosts data, thereby improving privacy [9].

On the other hand, NDN also brings new security challenges. One key challenge is that authenticity of data packets in NDN cannot be effectively verified by NDN routers since the packets may be from anywhere in the networks [11]. Therefore, an adversary can easily inject faked data packets or replay data packets, so as to poison content cache in NDN networks. In particular, cache poisoning can lead to denial of service (DoS)[1] [11]. Also, it is non-trivial to detect and throttle DoS attacks due to the flooding of fake data request packets in NDN. Our key observation is that NDN routers do not have any information about which content providers or users produce data packets. Therefore, NDN routers cannot readily decide if the packets in networks are malicious, although they parse the semantics of packets during packet forwarding.

This paper aims to fill the void in NDN by developing a security enforcement architecture based on *capabilities* [15], [23]. A capability serves like a "ticket" that specifies an access right to a data packet. Thus, we can enforce different security policies by embedding capabilities in packets. We can reject any unsolicited packet without a correct capability, thereby preventing potential DoS attacks triggered by fake or replayed packets. The use of capabilities also enables us to enforce security policies in a *distributed* manner, such that all NDN routers can verify the authenticity of any forwarded packet. While capabilities have been extensively studied in IP networks (e.g., [3], [4], [17], [18], [22]) for DoS/DDoS defense, deploying capabilities in NDN needs a fundamentally different design, mainly because of the unique in-network caching feature of NDN and any new attack that exploits this feature (e.g., the cache poisoning attack).

In this paper, we propose CSEA, a distributed capability-based security enforcement architecture tailored for NDN. We leverage existing security mechanisms in NDN with minimal extensions to implement and deploy a new capability mechanism. Specifically, we observe that the current NDN design requires mandatory digital signatures for all data packets, which we can leverage to verify the integrity and authenticity of the data packets by routers and end users [9]. By verifying the signature fields of data packets, all NDN routers and end users can verify if the data packets remain intact. We use this signature field to embed data packet

Manuscript received March 22, 2016; revised November 28, 2016; accepted May 1, 2017; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor S. Sen. Date of publication June 29, 2017; date of current version October 13, 2017. This work was supported in part by the National Key R&D Program of China under Grant 2016YFB0800102, in part by the National Natural Science Foundation of China under Grant 61572278, Grant 61402357, Grant 61572483, and Grant 61602457, and in part by the Research Committee of CUHK under Grant 3132964 and Grant 3132821. *(Corresponding author: Qi Li.)*

Q. Li is with the Graduate School at Shenzhen, Tsinghua University, Shenzhen 518055, China (e-mail: qi.li@sz.tsinghua.edu.cn).

P. P. C. Lee is with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Hong Kong (e-mail: pclee@cse.cuhk.edu.hk).

P. Zhang is with the Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China (e-mail: p-zhang@xjtu.edu.cn).

P. Su and L. He are with the Trusted Computing and Information Assurance Laboratory and the State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China (e-mail: supurui@tca.iscas.ac.cn; heliang@tca.iscas.ac.cn).

K. Ren is with the Department of Computer Science and Engineering, The State University of New York at Buffalo, Buffalo, NY 14260 USA (e-mail: kuiren@buffalo.edu).

Digital Object Identifier 10.1109/TNET.2017.2715822

[1]For simplicity, faked data injection and cache poisoning are collectively called data poisoning attacks in the rest of the paper.

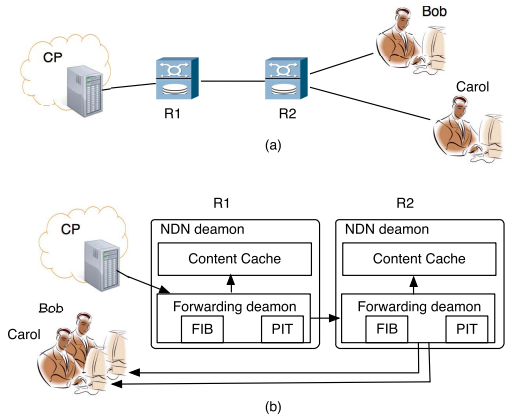

Fig. 1. Content delivery in NDN. (a) Sample topology. (b) The basic service flow in the NDN network.

capabilities that include the packet digest and the access right associated with a packet.

We further develop a *lightweight* one-time signature scheme based on the Merkle Hash Tree [20], [31] for CSEA. The one-time signature scheme utilizes standard hash functions to enable CSEA to quickly produce and verify data capabilities, which addresses the heavyweight problem in traditional public-key digital signature algorithms (e.g., RSA). CSEA generates, via hash functions, dynamic secrets, so as to guarantee the timeliness of packets and enable efficient verification of content authenticity. In particular, compared with the one-time signature scheme [20], [31] in the literature, our scheme significantly reduces the communication overheads.

To summarize, the contributions of this paper are three-fold:

- We propose CSEA, a capability-based security enforcement architecture that enables the verification of content authenticity in a *distributed* manner. CSEA can also throttle flooding-based DoS attacks.
- We develop a *lightweight* one-time signature scheme to ensure timeliness of packets and enable efficient verification of content authenticity.
- We implement our CSEA architecture on the CCNx platform, and evaluate its performance with testbed and Planetlab deployment. Our experimental results show that the overhead introduced by CSEA is negligible. In particular, CSEA only incurs around 4% delays in data packet retrieval.

II. NDN and Threats

A. NDN Overview

Named Data Networking (NDN) [14] is a content-based network architecture that delivers packets referenced by location-independent content names, as opposed to packet addresses as in traditional IP networks. NDN interconnects *users* (i.e., entities that request contents) and *content providers (CPs)* (i.e., entities that originate contents) through a set of content routers (or *routers* for short). It also supports *in-network caching*, such that content may be available at a CP or cached in the routers.

Figure 1 shows the basic workflow of an NDN network. To request specific content, a user issues an *interest* packet, which is propagated along the routers to the CP that holds the content. The CP then returns the content to the user in *data* packets. With in-network caching, if a router has cached the requested content, it stops forwarding the interest packet and directly returns the content to the user. Each router also keeps forwarded data packets in its local cache for later use.

Each router keeps two data structures for packet forwarding: *forwarding information base (FIB)* and *pending interest table (PIT)*. The FIB keeps a list of outgoing interfaces available in the router, while the PIT keeps the incoming interfaces of the received interest packets to allow data packets to be returned.

We assume that if both users and routers are benign, they follow the same NDN protocol for communications. We do not distinguish them and collectively call them *nodes* in the following discussion.

B. NDN Threats

NDN offers distinct security and privacy advantages over traditional IP network architectures through name-based transmission [9], [14]. On the other hand, NDN is vulnerable to *unsolicited* packets (including interest and data packets), such that adversaries can inject junk packets or replay outdated packets in order to disrupt the content delivery and in-network caching mechanisms of NDN. In this paper, we focus on three specific attacks as a result of the unsolicited packets.

Content Poisoning Attacks: An adversary can inject fake content to poison an NDN system. For example, an adversary can parse interest packets and inject fake content whose names match those of the interest packets. In particular, it can compromise a router and put the fake content in the local cache in advance, so that the fake content can be directly returned to the user.

DoS Attacks: An adversary can launch denial-of-service (DoS) attacks by flooding interest or data packets against the components of routers, such as the PIT and the local cache [14]. This denies the attacked routers from processing legitimate packets.

Content Leakage Attacks: Routers can cache data packets that are forwarded, and return the cached data packets directly to any router or user without restrictions. Thus, an adversary can retrieve content from upstream routers by sending or replaying unsolicited interest packets, even though it is not authorized to access the content.

C. Security Goals

To defend against the threats above, we aim for the following security goals:

- **Integrity:** Each interest or data packet remains intact.
- **Authenticity:** The content carried in each data packet indeed originates from the claimed CP.
- **Authorization:** A user who issues an interest packet is authorized to access the requested content.
- **Timeliness:** Each interest or data packet is not delayed and replayed by any malicious router.

Each user or router can verify and dismiss any received packet that violates any of the above security goals. Here, we do not consider the attacks that violate content confidentiality,

which can be readily enforced through encryption [9]. We also do not address the blackhole attacks sine NDN is internally resilient to them [28].

To make our discussion simpler, in this paper, we assume that each piece of content is delivered by a single data packet. We can also extend the methodology to deliver a piece of content in multiple packets (e.g., when the content size exceeds the maximum transmission unit (MTU) size), in which case each data packet includes an individual signature field.

III. CSEA Overview

In this section, we present a design overview of CSEA, a distributed capability-based security enforcement architecture for NDN.

A. Capabilities

CSEA associates each piece of content in NDN with a *capability*, which can be viewed as a ticket that specifies the access right of the content. In CSEA, capabilities serve two key purposes. First, capabilities enable a CP to authorize content under an access right, similar to the use of capabilities in classical computing systems [13], [23]. In addition, capabilities control the permission of how users access content, following the spirit of DoS-limiting networks [27].

In CSEA, a capability comprises two components: a *content signature* and a *token*. The content signature is the cryptographic digest of the content for ensuring the integrity and authenticity of the content, while the token encodes the access right of the content.

A token not only indicates the access right of the content, but also enables a user to subsequently access additional content of the same access right from the same CP. In other words, a token is CP unique, and can be only access a set of contents from the same CP. Any content will invalid tokens, e.g., faked or expired tokens, will not be verified and forwarded. In the meanwhile, a CP can embed a capability in the data packet that is to be returned to a user, meaning that the data packet is protected by the specified access right in the capability. When a router or the user receives the data packet, it checks the correctness of the capability. If the capability is valid, the data packet can be either forwarded (for a router) or delivered (for the user).

In addition, when a user receives the data packet, it can extract the token from the received capability and embed the token in an interest packet for accessing additional content from the same CP under the same access right. A router or the CP can verify the token with the one that it has previously observed. If the token is valid, then the user is authorized to access the content, and the interest packet will be either forwarded (for a router) or processed by the CP. Here, we assume that at the beginning, a user retrieves an initial token from a CP through registration. A user does not need to explicitly retrieve and update tokens. Instead, tokens will be automatically updated once the contents are retrieved from the content provider and verified.

Note that tokens and capabilities can be used to verify each other in each NDN node. The node will drop any invalid tokens or capabilities that cannot be successfully verified. Therefore, it is not possible to construct capability forgery attacks.

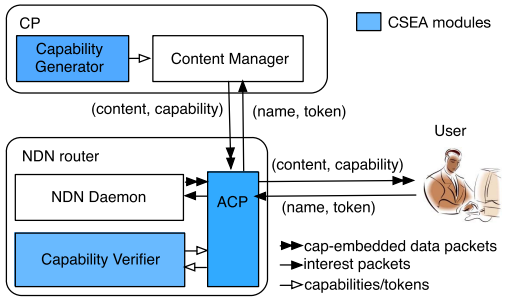

Fig. 2. Capability-based security enforcement architecture (CSEA).

B. CSEA Architecture

Figure 2 depicts the architecture of CSEA and how it is integrated into the existing NDN design. Each CP includes a *capability generator*, which generates capabilities for outgoing data packets. Each router includes a *capability verifier*, which verifies the correctness of capabilities from CPs and tokens from users, and an *access control point (ACP)*, which enforces access control policies and decides whether an interest or data packet should be forwarded or dropped based on the verification results of the capability verifier. The ACP also caches received tokens that are deemed valid, and uses them to verify the subsequently received capabilities and tokens. Moreover, it counts and limits the times of valid tokens that are used to request different data packets so as to prevent interesting flooding with valid tokens.

C. Open Issues

By verifying capabilities, CSEA ensures integrity and authenticity of data packets, thereby achieving data access authorization. On the other hand, traditional public key signature schemes, such as RSA and DSA, are heavyweight and cannot ensure timeliness of packets [11], and thus cannot be used for our capability generation and verification. To this end, our CSEA design aims to address the following two issues:

a) Lightweight capability management: Given that routers typically have limited resources, CSEA should introduce a small computation and communication overhead of capability generation and verification, such that it will not significantly degrade the data forwarding performance of an NDN system.

b) Freshness of tokens: In CSEA, users use tokens to access authorized content, while routers use tokens for verification and access control. Thus, CSEA should support efficient token revocation and refreshment, so as to avoid the replay of tokens.

IV. Lightweight Capability Management

In this section, we present the design details of how CSEA achieves lightweight capability generation and verification, inspired by the one-time signature (OTS) algorithm [16], [20], [31].

A. Bootstrapping

Before accessing content from a CP, a user needs to first obtain an initial token from the CP through a secure channel (e.g., by leveraging a secure content dissemination scheme designed for ICN [26]). Specifically, the CP first constructs a *secret key vector* $\mathbf{x} = (x_1, x_2, \cdots, x_l)$, where x_i $(1 \leq i \leq l)$ denotes a secret random number and l is the length of $\mathbf{x}$. Here, l is a configurable parameter that determines the size of the corresponding capability (see Section IV-C). It then generates a token $t = h(f(h(x_1))||f(h(x_2))||\cdots||f(h(x_l)))$, where $h(.)$ is a cryptographic hash function used for token generation, $f(.)$ is a truncation function that extracts the lower l-bits of the input hash value, and $||$ is the concatenation operator. The CP then returns t to the user through the secure channel.

Each initial token is associated with an access right. If the user accesses content of a different access right from the same or a different CP, it needs to obtain another initial token. For brevity, the following discussion assumes that all requested contents have the same access right.

B. Requesting Content With Tokens

To access content from a CP, a user issues to the CP an interest packet that is embedded with the associated token t. The interest packet traverses along a set of routers, each of which will verify the embedded token t with its own cached tokens. Specifically, each router maintains two sets of cached tokens: the *request token set* $\mathcal{T}_Q$ and the *response token set* $\mathcal{T}_P$, which store the valid tokens embedded in an interest packet and a data packet, respectively. If $\mathcal{T}_P$ is empty (right after the bootstrapping stage) or t matches one of the tokens in $\mathcal{T}_P$ (the latter means that the same token has been correctly observed in the last data packet), then t is considered valid. The router will store t in $\mathcal{T}_Q$ and forward the interest packet until the interest packet reaches the CP or any router that caches the content. Otherwise, if t is invalid, the interest packet will be dropped. $\mathcal{T}_Q$ will be used to verify any returned capability later, which we further discuss in Section IV-D.

If a router has already cached the requested data packet, it verifies whether the token embedded in the interest packet has already been stored in $\mathcal{T}_P$. If so, the router directly returns the requested data packet; otherwise, it drops the interest packet. We discuss this operation in detail in Section IV-E. Moreover, as we discussed above, $\mathcal{T}_P$ plays a key role in verifying tokens. Normally, token verification with $\mathcal{T}_P$ does not pose a problem of false negative since any token not in $\mathcal{T}_P$ will be accurately dropped. However, the false positives of token verification may impact the accuracy of interest verification and further influence the accuracy of capability verification. We will analyze the false positives of token verification with $\mathcal{T}_P$ in Section V-C.

C. Capability Generation

CSEA utilizes the Merkle Hash Tree (MHT) algorithm to efficient generate capabilities for various content packets. When a CP receives an interest packet embedded with a correct token t from a user, it generates a capability for the requested content. First, it constructs a bit vector $\mathbf{g}$, which will later be used for capability generation. The bit vector $\mathbf{g}$ is formed by the MHT algorithm [20] (see an example of how the MHT is formed in Figure 3). Specifically, the CP generates a new secret key vector $\mathbf{x}' = (x'_1, x'_2, \cdots, x'_n)$ and hence a new token $t' = h(f(h(x'_1))||f(h(x'_2))||\cdots||f(h(x'_l)))$, which is similar to that in the bootstrapping stage (see Section IV-A). The CP will split the content (denoted by C) into multiple parts to efficiently compute its capability. Given the new token t' and the multiple data parts of C (see Figure 3), the CP constructs a binary MHT, in which each leaf node is the hash value of either the token t' or a data part, and each non-leaf node is the hash value of the concatenation of the hashes of the left and right child nodes. The CP then computes l-bits value according to the hash value at the root to form the bit vector $\mathbf{g}$ via the $k(.)$ function. $k(.)$ is a concatenation function that concatenates the lower (l-$\lfloor log_2{}^l \rfloor$-1)-bits of the input hash value with ($\lfloor log_2{}^l \rfloor$+1)-bits that count the number of '1' bits in the lower (l-$\lfloor log_2{}^l \rfloor$-1)-bits. By counting the number of '1' bits, $k(.)$ effectively prevents an adversary from constructing capabilities with $\mathbf{g}$ that is computed from fake content [20], [31] and can be verified.

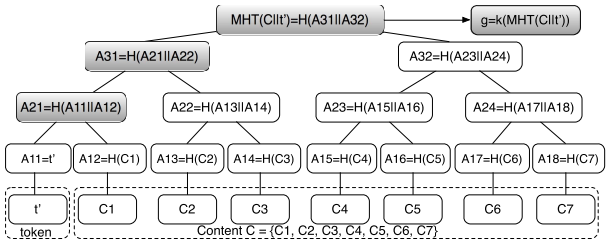

Fig. 3. Computation of $\mathbf{g}$ with the MHT algorithm. Here, the content C is split to seven physical packets, i.e., $C_1, C_2, \cdots, C_7$. The MHT is formed by the token t' and the set of data packets.

In CSEA, we use the MHT algorithm to reduce the delay of capability generation. Our observation is that the same content in NDN may be distributed to different users, each with a different secret. Also, the same user may request the same content at a later time, yet the capability may expire and need to be updated. When CP reconstructs a new capability, it simply needs to update the token, as well as the corresponding leaf node and all non-leaf nodes along the path to the root in the MHT (see the shaded nodes in Figure 3). Here, the CP does not need to generate hashes for the content again and thereby capability generation and verification is faster than pure hash functions. Our evaluation shows that using the MHT algorithm reduces the delay of capability generation over 60% compared to using pure hash functions on the token and the entire content (see Section VII).

Algorithm 1 shows the pseudo-code of the capability generation algorithm. The inputs include the requested content C, the new token t', and the original secret key vector $\mathbf{x}$ associated with the token t embedded in the interest packet. First, the algorithm computes the bit vector $\mathbf{g} = (g_1, g_2, \cdots, g_l)$ of length l through MHT as discussed above (step 1) according to

C and t. It computes the signature (denoted by S) by checking the j-th bit g_j of $\mathbf{g}$ (where $1 \leq j \leq l$) as follows (see steps 2-8). If $g_j = 0$, then it sets $s_j = f(h(x_j))$ (step 4); otherwise if $g_j = 1$, it directly sets $s_j = x_j$ (step 6). Finally, it returns the signature S by concatenating all s_j's (step 9).

The CP then constructs a data packet that includes content C and the capability, where the latter comprises two components: the content signature S and the new token t' (see Section III-A). It returns the data packet to the user requesting the content.

Algorithm 1 Produce

Input: Data packet C, token t', key vector $X = \{x_1, x_2, \cdots, x_n\}$;
Output: Capability(S, t');
1: $g \leftarrow k(\mathrm{MHT}(C||t'))$;
2: **for** $(j = 1 \rightarrow l)$ **do**
3: **if** $(g_j = 0)$ **then**
4: $s_j \leftarrow f(h(x_j))$;
5: **else**
6: $s_j \leftarrow x_j$;
7: **end if**
8: **end for**
9: $S \leftarrow s_1||s_2||\cdots||s_l$;

D. Capability Verification

Capability verification is similar to capability generation. When a router receives a data packet, it verifies the embedded capability based on its cached tokens in $\mathcal{T}_Q$ (see Section IV-B). Algorithm 2 shows the pseudo-code of the capability verification algorithm. It first computes the bit vector $\mathbf{g}$ with the MHT algorithm based on the received data packet C and the embedded token t' (step 1). It then uses the signature S to compute the verification tag V: if the j-th bit of $\mathbf{g}$ (where $1 \leq j \leq l$) is $g_j = 1$, it sets $v_j = f(h(s_j))$ (step 4); otherwise if $g_j = 0$ it sets $v_j = s_j$ (step 6). It concatenates all v_j's to obtain V (step 9). Finally, the algorithm returns true if V exists in $\mathcal{T}_Q$ (steps 10-14), or false otherwise.

Also, the router caches both V and t' in $\mathcal{T}_P$ if they are not in $\mathcal{T}_P$, so it can validate the token embedded in the next interest packet (see Section IV-B). We provide more details how $\mathcal{T}_P$ is used in the next subsection.

E. Caching Tokens From Data Packets

Recall that $\mathcal{T}_P$ is used to cache tokens received from data packets. In addition to authenticating new interest packets (see Section IV-B), we discuss more usage of $\mathcal{T}_P$.

Freshness of Tokens: CSEA leverages $\mathcal{T}_P$ to ensure the freshness of tokens by associating each token with a limited valid time period, such that it can revoke any token from $\mathcal{T}_P$ after the token has stayed in $\mathcal{T}_P$ for the time period. If a received token does not exist in a router's $\mathcal{T}_P$, it means that the token expires and will be dropped by the router (see Section IV-B). Since any verified token embedded in incoming data packets are stored in $\mathcal{T}_P$ (see Section IV-D), tokens in $\mathcal{T}_P$ are kept being refreshed through capability verification.

Algorithm 2 Verify

Input: Data packet C, capability $(S = s_1||s_2||\cdots||s_l,\ t^\dagger)$, request token set $\mathcal{T}_Q$;
Output: True: success; **False:** failure;
1: $g \leftarrow k(\mathrm{MHT}(C||t^\dagger))$;
2: **for** $(j = 1 \rightarrow l)$ **do**
3: **if** $(g_j = 1)$ **then**
4: $v_j \leftarrow f(h(s_j))$;
5: **else**
6: $v_j \leftarrow s_j$;
7: **end if**
8: **end for**
9: $V \leftarrow h(v_1||v_2||\cdots||v_l)$;
10: **if** $(V \in \mathcal{T}_Q)$ **then**
11: return **true**;
12: **else**
13: return **false**;
14: **end if**

Timeliness of Packets: CSEA verifies tokens via $\mathcal{T}_P$, and in turn ensures timeliness of both interest and data packets. Specifically, if a router receives an interest packet whose embedded token does not exist in $\mathcal{T}_P$, it will drop the interest packet to prevent the packet from retrieving any data packet. Similarly, if a router receives an expired data packet, it can verify that the embedded capability in the data packet does not match any token in $\mathcal{T}_Q$ (see Section IV-D), which contains valid tokens that are verified by the tokens stored in $\mathcal{T}_P$ (see Section IV-B), and the router will drop the data packet as well. In short, CSEA will deny any interest packet with an expired token or any data packet with an expired capability.

V. Analysis

A. Security of CSEA Capability Scheme

CSEA leverages MHT to produce a token t and signs data packets with the signature scheme (t, h), where t is the generated token and h is a collision-resistant hash function. We can obtain the following theorem and prove it by showing that any polynomial time algorithm breaks the unforgeability of the proposed signature scheme only with negligible probability in the random oracle model [5], [16].

Theorem 1: Given a collision-resistant hash function h and a token t, the proposed signature scheme $(t,\ h)$ in CSEA is unforgeable.

Proof: Without the loss of generality, we assume that a key vector generating a token is $X = \{x_1, x_2, \ldots, x_n\}$ and the generate token is $t = h(f(h(x_1))||f(h(x_2))||\ldots||f(h(x_n)))$, and the adversary is intending to forge a signature scheme corresponding to the key vector X.

Suppose that the adversary presents (C', S'), where C' is the data packet, and S' is the forged capability (or signature) corresponding to C'. Given data packet C' with the token t, CSEA generates the signature S. Therefore, we are interested in computing the probability that $S = S'$ given C'. In other

words, the adversary makes the output of signature verification true if and only if $S = S'$.

In order to achieve $S = S'$, the adversary has to construct $\mathbf{g}$ that is used to produce S' for C' and compute $s_j = s'_j$ given the constructed $\mathbf{g}$. The probability of generating $\mathbf{g}' = \mathbf{g}$ is $\frac{1}{2^l}$ where l is the length of $\mathbf{g}$. Even if the adversary successfully constructs $\mathbf{g}' = \mathbf{g}$, the adversary has to construct $s_j = s'_j$ for each g'_j. For those $g'_j = 0$, the adversary has to correctly compute $s'_j = f(h(x_j))$, where x_j is unknown to the adversary. Hence, the probability of generating output $s'_j = f(h(x_j))$ by adversary is $\frac{1}{2^l}$ where h is modeled as a random oracle. In addition, for g'_j where $g'_j = 1$, the adversary has to correctly compute $s'_j = x_j$, where x_j is unknown to the adversary. Therefore, the probability of generating output $s'_j = x_j$ is $\frac{1}{2^{|x_j|}}$. Since $|x_j| \geq l$, the probability of generating output $S' = S$ by the adversary for constructed $\mathbf{g}'$ is less then $(\frac{1}{2^l})^l$. Therefore, we can obtain that the overall probability of generating $S' = S$ is less than $(\frac{1}{2^l})^{l+1}$.

Therefore, we can conclude that $(\frac{1}{2^l})^{l+1}$ is negligible, where $l = 32$. Therefore, the probability of breaking the unforgeability of the signature scheme by the adversary is negligible. ■

B. CSEA Security Properties

In this section, we briefly analyze the security properties achieved by CSEA and discuss how CSEA counters sophisticate attacks.

Data Poisoning Attacks: Each data packet piggybacks a capability (S, t') such that each NDN node can verify the data by verifying (S, t') and tokens in valid token array $\mathcal{T}_P$. Any fake or malicious data packets cannot be verified and will be dropped.

DoS Attacks: Routers can leverage tokens cached in their $\mathcal{T}_P$ to verify interest and data packets. Under DoS of flooding data packets, only data packets with verified capabilities can go through a network if they are requested by the users in the network. Fake data packets, data packets with expired capabilities, or any unsolicited data packets cannot be successfully verified by routers so that they will be dropped. Hence, CSEA can effectively throttle DoS attacks by flooding data packets.

Moreover, CSEA defends against DoS attacks by flooding interest packets. Each router maintains $\mathcal{T}_P$ and verifies if an incoming token t is valid by comparing it with tokens in $\mathcal{T}_P$. Routers can infer which $\mathcal{T}_P$ can be used to verify tokens in the interest packets according to the embedded hierarchy names in the packets [14]. Therefore, under DoS attacks of flooding interest packets, malicious routers cannot simply replay or fake tokens to generate interest packets. Any packets with invalid tokens will be dropped since they cannot match the records in $\mathcal{T}_P$. Therefore, CSEA can effectively mitigate DoS attacks by flooding fake or expired packets. In addition, CSEA counts the number of times that tokens are used to request data packets, especially the data packet referring to non-existing content, and then throttles the interest flooding attacks with valid tokens by limiting the usage of the tokens. It can also defend against the content flooding attacks that are constructed by flooding interest packets with valid tokens. In order to successfully construct the attacks, the attackers need to collect various valid tokens, which is difficult to achieve.

Note that, during network bootstrapping, routers may not be able to detect the DoS attacks mounted by flooding interest packets. At this stage, CPs have not sent out any data packets to the network yet, and then the routers do not cache any valid tokens in their $\mathcal{T}_P$ for token verification. Thus, malicious interest packets may still be delivered by the routers. The worst case here is that the malicious packets will arrive in the CP and the CP drops them. However, the malicious packets will be dropped once the data packets are produced and propagated in the network. To efficiently defeat the attacks during network bootstrapping, we can allow routers to cache the initial tokens from CPs in their $\mathcal{T}_P$ during network bootstrapping.

Replay Attacks: Each router only maintains a set of valid tokens within a period. Tokens are periodically refreshed in the router even when the same data traverses it. Any replayed interest and data packets cannot successfully go though it because the embedded tokens or capabilities cannot be verified with the refreshed tokens. In particular, a malicious node cannot retrieve any data by replaying interest packets since the embedded tokens cannot be verified. Moreover, since each router counts and limits the times of valid tokens that are used to request different data packets, the impact of replaying valid tokens is limited. Therefore, CSEA can mitigate interest and data replay attacks.

Content Leakage Attacks: CSEA verifies if token t in request token set $\mathcal{T}_Q$ is authorized to access the data packet by verifying t with capability (S, t') in the data packet. Any fake token that can not be successfully verified will be dropped. Hence, the data packet will not be leaked out to unauthorized users without valid tokens. However, data packets may be still leaked out by malicious nodes that do not comply with CSEA, i.e., they do not verify tokens in $\mathcal{T}_Q$ but directly return the data packets to the requesters. However, the effect on the attacks is very limited. Since the nodes along the path from the malicious nodes to the destination nodes do not record interest packets for the incoming data packets, they will drop the unwanted packets. Any data packets will be dropped if there does not exist any valid tokens $\mathcal{T}_Q$ to request the data packet. The attacks will succeed only when CSEA on all nodes along the forwarding path is tampered with to opt in to collude. In this scenario, data packets may be leaked by the malicious nodes. Nevertheless, we believe that the scenario is very rare in real practice.

C. Analysis of False Positives in Verifying Tokens

We also analyze the impact of the size of the token response set $\mathcal{T}_P$ on false positives in verifying tokens (see Section IV-B). Since $\mathcal{T}_P$ can only keep a finite number of tokens with limited memory, a valid token will be falsely dropped and cannot be authorized to retrieve data packets if the token does not match any record in $\mathcal{T}_P$, which raises false positives in verifying tokens. In the following analysis, for simplicity, we consider the case where there are m users connected to one router R and users request a set of content, i.e., C, from the same CP. However, the analysis results can be applied to the scenarios

where content is from multiple CPs and users are connected with different routers. In CSEA, after a user retrieves a data packet, the token will be updated in the routers delivering the packets. The size of $\mathcal{T}_P$ is set to n, which means that R can hold up to n tokens for C. Initially, the token set $\mathcal{T}_P$ is empty, and the oldest token is removed from the chain if $\mathcal{T}_P$ is full. Here, we assume users are independent in requesting C, and, for each user, the inter-arrival time of interest packets is memoryless. Thus, the arrival of interest packets from user u_i follows a Poisson process. Let the arrival rate of the interest packet from user u_i be λ_i.

Now we consider the instant when u_i gets data after issuing an interest packet. According to the design of CSEA, u_i would also obtain a token, t. Suppose that u_i would issue the next interest after a period of τ. Let $N_{\setminus i}(\tau)$ be the interests issued by the other $m-1$ users. If $N_{\setminus i}(\tau) \leq n$, then the token t would be removed from the tail of $\mathcal{T}_P$, and the interest packet issued by u_i would be treated as "malicious" one and dropped falsely. Here, we aim at analyze this false positive rate P_{fp}.

Let T_i be the inter-arrival time of u_i's interests, then P_{fp} can be calculated as:

$$P_{fp} = \int_0^\infty \Pr\{N_{\setminus i}(\tau) > n | T_i = \tau\} f_{T_i}(\tau) d\tau \tag{1}$$

As the other $m-1$ users are independent, $N_{\setminus i}(\tau)$ follows a compound Poisson process with parameter $\lambda = \sum_{j \neq i} \lambda_j$. Let S_n be the time when the n-th interest arrives from the $m-1$ users, we can obtain that:

$$\begin{aligned} \Pr\{N_{\setminus i}(\tau) > n\} &= \Pr\{S_n \leq \tau\} \\ &= \int_0^t \lambda e^{-\lambda x} \frac{(\lambda x)^{n-1}}{(n-1)!} dx \\ &= \int_0^\lambda \tau e^{-\tau x} \frac{(\tau x)^{n-1}}{(n-1)!} dx \end{aligned} \tag{2}$$

Thus, we can compute the false positive rate as:

$$P_{fp} = \int_0^\infty \int_0^{\sum_{j \neq i} \lambda_j} t e^{-tx} \frac{(\tau x)^{n-1}}{(n-1)!} dx \lambda_i e^{-\lambda_i \tau} d\tau \tag{3}$$

$$= (1 - \frac{\lambda_i}{\sum_j \lambda_j})^n \tag{4}$$

Based on the above analysis, we can conclude that the false positive rate drops exponentially with the increase in n, i.e., the size of $\mathcal{T}_P$.

D. Discussion

Effectiveness of Replay Defense: CSEA significantly increases the cost of launching the replay attacks by limiting the validity time of capabilities, and hence mitigates the replay attacks. However, a shorter expiration time will incur a side effect of increasing the communication overhead. We have analyzed false positives of packet verification by a short expiration time in Section V-C. We show that the false positive rate drops exponentially with the increase in the size of the response token set $\mathcal{T}_P$. Actually, the side effect of the false positive rate is also impacted by other factors, e.g., content cache expiration time and network topologies. We pose the theoretic analysis on the side effect as future work.

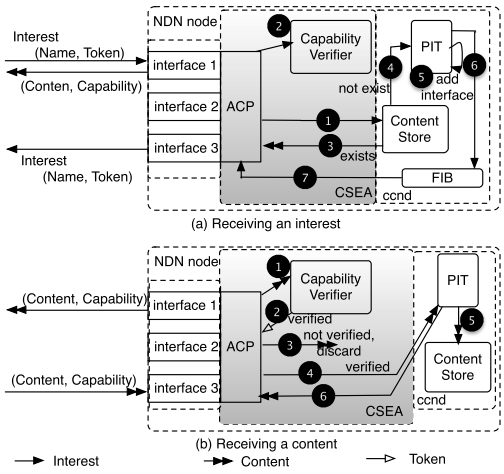

Fig. 4. CSEA implementation in NDN.

Collusion Attacks: In this paper, we assume that content producers are benign. Actually, we cannot easily infer if a content provider produces bogus data. The problem is similar to the current Internet. In order to address this issue, we can leverage trust schemas [30] in NDN to link CP's authentication keys and trust rules for their content.

Deployment on Other ICN Designs: CSEA can be applied to other ICN architectures similar to that presented in this paper. We can achieve the goal by piggybacking tokens in content request packets and capabilities in data packets. Any router can verify tokens and capabilities to ensure the security of the ICN architecture.

VI. Implementation

We implement CSEA with thousands LOC in the NDN prototype called CCNx [1] that is developed by Xerox PARC. CCNx provides a daemon program called *ccnd* including Content Store, PIT, and FIB. The daemon forwards interest packets according to FIB, and deliveries contents according to the entries in PIT, and stores content in Content Store. (see Figure 4). Also, CCNx can be set up as a client to deliver requested data packets to user applications. CCNx prototype will cache all data packets it delivered. CSEA introduces an extension to NDN such that NDN has abilities to defend against different attacks discussed in Section II: (i) CSEA reuses digital signature fields of data packet to piggyback capabilities and extend interest packets to embed tokens. (ii) CSEA verifies each packet before the packet is delivered to the ccnd daemon.

Figure 4 shows interest and data processing procedures in the CSEA prototype. The ACP module in CSEA plays a key role in enforcing capability-based policies on all interfaces. As we discussed in Section IV-D, in order to accurately enforce the policies in ACP, CSEA also includes the capability verifier to engage in verifying user tokens and data capabilities. The capability verifier verifies tokens in interest packets that requests cached data packets (see Figure 4(a)), and verifies tokens and capabilities (see Figure 4(b)) upon incoming data packets.

When an interest packet arrives, ACP will verify the token embedded in the interest packet and check if the token matches a record in $\mathcal{T}_{\mathcal{P}}$ that is stored in ACP (see Figure 4(a)). ACP forwards the interest packet to the ccnd daemon if the token is verified (see arrow (1)). Meanwhile, it will forward the token to the capability verifier as well and is stored in $\mathcal{T}_{\mathcal{Q}}$ in the capability verifier (see arrow (2)). Later the capability verifier can verify the validity of the stored tokens when the requested data packet is received. If the requested data packet is already cached, ACP will directly decide if the stored tokens in ACP are allowed to retrieve the data packet (see arrow (3)). If the node does not cache the data packets or does not receive the interest packet before, ACP will allow the interest packet to be further propagated (see arrow (7)).

Upon receiving a data packet, as shown in Figure 4(b), ACP will forward the capability to the capability verifier for data verification (see arrow (1)). The capability verifier will return the valid tokens to ACP according to the token in $\mathcal{T}_{\mathcal{Q}}$ after successful data verification (see arrow (2)), and the returned tokens will be used for verification of new incoming interest packets. ACP will drop the data packet if the set of returned tokens is empty, which means that the received data packet does not include correct capability or the tokens requesting the data packet is not authorized to retrieve the data packet (see arrow (3)). In this setting, ACP can accurately authorize different interests to retrieve the data packet according to the valid tokens. In the meanwhile, the verified data packet will delivered to the ccnd deamon (see arrow (4)). For outgoing data packets, ACP will allow the packets to be delivered to an interface if and only if its ACP has a record associated with the interest packet sent from the interface (see arrow (6)).

VII. Performance Evaluation

In this section, we evaluate the performance and overhead of CSEA. We focus on the experimental data of the CSEA prototype with microbenchmark and macrobenchmark experiments. We deploy the prototype in a small-scale testbed and Planetlab, which aims to demonstrate that CSEA incurs small computation and communication overheads while enhancing the security of NDN. The hash function used in CSEA is SHA-256. Note that, CSEA can produce a capability for a data packet that is splitted into more than one physical packets before it is delivered into the network. In the experiments, we set the sizes of data packets between 100 bytes and 8K bytes[2] The packets will segment when the packet sizes are larger than 1.5K bytes.

A. Performance in Testbed Deployment

In the testbed experiments, we study different performance aspects of CSEA with different parameters. Our CSEA testbed is a line topology composed of three nodes deployed on an Intel Core i5 3470 machine with 3.2 GHz that runs Linux.

[2]As suggested in the NDN design [14], content will be split into different smaller-size packets (called trunks). Therefore, NDN will not generate large packets. Actually, according to our experiment results, our scheme will not introduce significant overhead in packet delivery even if there are packets more than 8K bytes. For simplicity, we only present the experiment results with content less than 8K bytes.

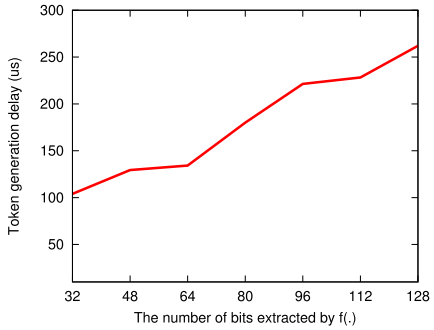

Fig. 5. Experiment 1: Token generation delays different lengths of tokens.

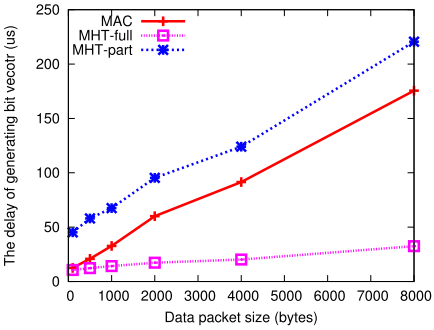

Fig. 6. Experiment 2: Bit vector generation delays with different hash functions.

Experiment 1 (Token Generation Delay): In this experiment, we evaluate token generation delays with different sizes of secrets. With different sizes of secrets, $f(.)$ extracts the equivalent length of bits from the generated hash values to generate tokens. We generate tokens with different sizes of secrets five times and measure the average delays. As shown in Figure 5, we observe that the token generation delays increase with the increase in the size of secrets since the number of hash function operations increase. The maximum delay with 128 secrets is around 250 μs. The difference between token generation delays with different secret sizes is bounded by 150 μs. Overall, the delay increase around 17% when the size of secrets increase 16. Note that, if we choose a larger size of secrets to generate tokens, it will increase the communication overhead because the sizes of capabilities embedded in data packets is decied by the size of the secrets (see Section IV-C). Therefore, in the rest experiments, we will set the size of secrets to 32.

Experiment 2 (Capability Generation and Verification Delay): We now evaluate the capability generation and verification delays with different sizes of data packets. As shown in the capability generation and verification algorithms (see Section IV-C and IV-D), $\mathbf{g}$ plays a key role in capability generation and verification. We firstly evaluate and compare the delays of big vector $\mathbf{g}$ generation with different packet sizes. Figure 6 depicts the experiment results. We observe that, compared with pure hash functions, the delays of generating the bit vector with incremental MHT construction (or "MHT-part" in short) are significantly reduced by more than 60%, though full MHT construction (or "MHT-full"

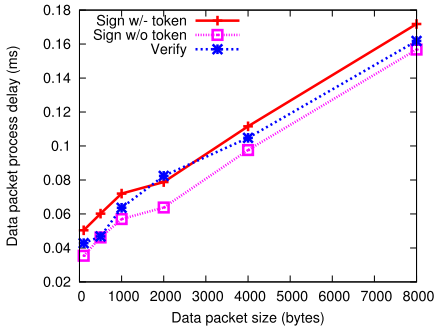

Fig. 7. Experiment 2: Capability generation and verification delay with different content.

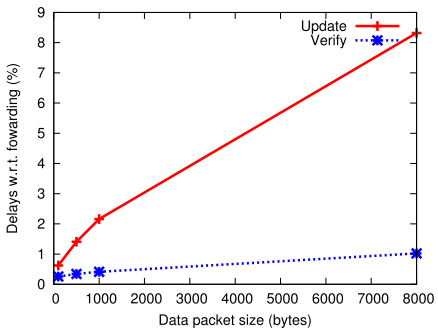

Fig. 8. Experiment 2: Computation costs with different sizes of content.

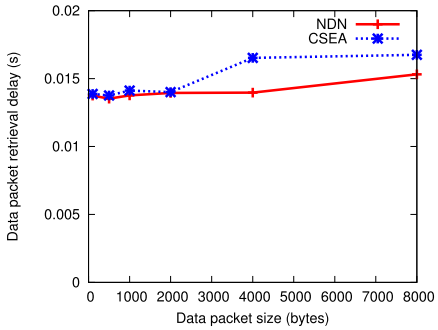

Fig. 9. Experiment 3: New data retrieval delay with and without CSEA.

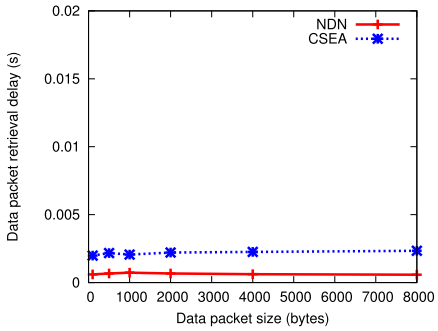

Fig. 10. Experiment 4: Cached data retrieval delay with and without CSEA.

in short) introduces more delays than pure hash functions (or "MAC" in short). In particular, the MHT-part approach incurs only around 32.5 μs to generate $\mathbf{g}$ for 8K bytes content, while the MAC approach takes more than 175 μs. The reason is that as opposed to MAC that operates on long messages, MHT operates on a small number of shorter messages with less computation overhead and hence incurs shorter delays. Therefore, it is more efficient to produce and verify capabilities with MHT in NDN.

We secondly evaluate the delays of capability generation and verification. Figure 7 illustrates experiment results. It is not surprising that capability generation delays increase with the increase in packet sizes since the times of hash function operations increase when packet sizes increase. The average capability generation delays are around 0.076 ms, and the average capability verification time is about 0.084 ms. Since the token generation delays are really small, if the capability generation procedure includes the token generation procedures, we can observe that the overall capability delays are still small and the value is around 0.091 ms. The delays are still acceptable though they includes around 15% more delays.

Moreover, we measure and compare the delays of capability verification and token updates. As shown in Figure 8, token updates incur only less than 8% of additional forwarding delays, while capability verification incurs only less than 1% of additional forwarding delays. Therefore, the cost of token updates is acceptable for content verification and content access control. We conjecture that the overall delays can be further significantly reduced if we can leverage hardware to implement hash functions.

Experiment 3 (New Data Packet Retrieval Delay): We now compare the delays to retrieve new data packets among three nodes (see Figure 9). New data packets mean that a user retrieve data packets from CPs, and the intermediate nodes did not deliver the request data packets before, which means that they do not cache the data. We can observe that the delays incurred by CSEA are really small. To verify capabilities with 100 bytes data packets, the packet retrieval delays with and without CSEA are 13.8 ms and 13.7 ms. Similarly, the delays to verify capabilities with 8K bytes data packets are around 15.3 ms and 16.7 ms. With different data packet sizes, the average packet retrieval delays are 10.5 ms and 11.1 ms, respectively. Averagely, CSEA introduces only 0.6 ms delays to retrieve new data packets compared with the naive NDN prototype. The increase rate of data packet retrieval delays is about 0.05%. Based on the observations, we can conclude that NDN introduces negligible overhead for a user to retrieve new data packets from CPs.

Experiment 4 (Cached Data Packet Retrieval Delay): In this experiment, we evaluate the delays to retrieve cached data packets. By caching data packets, the node requesting data packets can directly obtain the data packets from the intermediate nodes. Figure 10 shows the retrieval delays of cached data packets. The packet retrieval delays are around 0.6 ms. CSEA introduces relatively constant packet processing delays that are around 1.5 ms. Compared to the delays of retrieving new data packets, CSEA incurs relatively more delays in retrieving data packets and introduces around 0.1% delays. By retrieving cached data packets, the delays both with and without CSEA are significantly reduced,

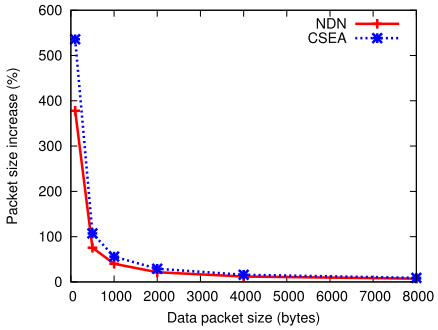

Fig. 11. Experiment 5: Communication overhead incurred by piggybacking capabilities.

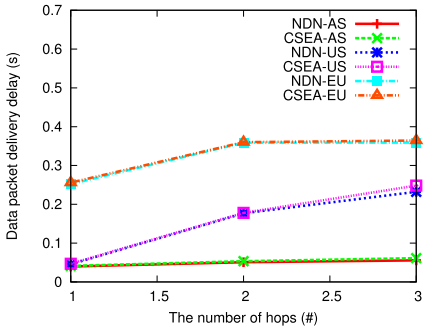

Fig. 12. Experiment 6: Data packet retrieval delays on Planetlab across different continents.

and the average reduction rates are around 96% and 86%, respectively. We observe that the overhead is still small and acceptable. Moreover, in the current prototype, we implement an array to cache different tokens in ACP and the capability verifier. The performance would be significantly improved if the token cache is implemented with hash tables, e.g., bloom filters. We leave an enhanced implementation of CSEA to future work.

Experiment 5 (Communication Overhead): We now evaluate the communication overhead incurred by CSEA. Since CSEA will piggyback tokens in interest packets and capabilities in data packets, it will increase the size of NDN packets. Figure 11 illustrates the communication overheads with different data packet sizes. We compare the packet size increase rate with native NDN and CSEA. As we discussed in Section III, each capabilities comprises a data signature and a token. In this experiment, for simplicity, we only evaluate the overhead incurred by piggybacking capabilities in data packets. For small packets, for example, both NDN and CSEA incur more than 2 times of the original data packet sizes. However, with the increase in packets size, the increase rate of packet sizes is significantly reduced. In average, NDN introduces extra 16% packet size to piggyback data signature, and the packet sizes with CSEA increase 22%. Therefore, CSEA only slightly increases the NDN packet sizes.

B. Performance in Planetlab Deployment

To evaluate CSEA in real production networks, we deploy the CSEA prototype on Planetlab. We deploy CSEA on ten Planetlab nodes across three continents, i.e., America, Asia, and Europe. For simplicity, we only measure the packet retrieval delays between three node pairs in the same continents. These nodes can reach each other directly, by one intermediate node in the other continent, or by two intermediate nodes in the other continent. Therefore, the paths between these node pairs are evaluated with different hop numbers, i.e., one, two, and three hops.

Experiment 6 (Data Packet Retrieval Delay in Planetlab): In this experiment, we measure the delays to retrieve data across different paths on Planetlab. Figure 12 shows the data packet retrieval delays with CSEA versus that with native NDN. We do not observe any significant delays incurred by CSEA. The packet retrieval delays between two direct neighbors with CSEA are almost the same to that without CSEA. The data packet retrieval delays across one intermediate node with and without CSEA are around 0.64 seconds and 0.66 seconds, respectively, and the increase rate is bounded by 0.03%. If the data retrieval paths include two intermediate nodes across different continents, the paths between America and Europe across Asia incur the smallest delays to retrieve data packets. The delays with different paths across Asia, America, and Europe, averagely increase 1%, 4%, and 6%, respectively. Note that, if data retrieval paths are more than two hops, the data packets may be retrieved from CP nodes or intermediate nodes caching the data. These results illustrate that CSEA introduces stable delays in retrieving data packets. On average, it only introduces 0.02 seconds delays to retrieve a packet. Compared with native NDN, the average increase rate of packet retrieval delays are around 4%. We believe that the packet processing overheads incurred by CSEA will be negligible if it is deployed with large-scale deployments on the Internet.

VIII. Related Work

Capability-based systems have been extensively studied in the literature [3], [4], [15], [17], [18], [22], [23]. For example, Shapiro *et al.* [23] develop a capability system to enforce resource access control within operating systems. Anderson *et al.* [3] propose to defend against DoS attacks in IP networks by issuing capabilities for packet sources. In contrast, CSEA targets an NDN system and aims to achieve both data access control and DoS defense through capabilities.

Access control enforcement schemes have been proposed for Information Centric Networking (ICN) (of which NDN is an instance). Fotiou *et al.* [10] propose a centralized access control architecture for ICN, in which access control providers issue credentials for data requests. Li *et al.* [16] propose LIVE, a centralized access control enforcement scheme for NDN. While LIVE also leverages Merkle Hash Trees as CSEA for lightweight data verification, [3] the fundamental difference between LIVE and CSEA is that LIVE enforces access control policies in a centralized manner, while CSEA

[3]Actually, the LIVE verification algorithm is vulnerable to the guessing attacks that adversaries can construct verifiable capabilities with g that is computed from fake content.

allows distributed access control. LIVE requires CPs to manage token distribution for access control, and hence routers and users require frequent interactions with CPs to obtain tokens. In contrast, CSEA distributes the load of data verification among NDN routers. Also, CSEA effectively throttles DoS attacks, which are not considered by LIVE.

DoS/DDoS countermeasures have been considered in NDN [8], [11]. They mitigate DoS attacks by identifying and suppressing malicious traffic. In contrast, CSEA can detect malicious traffic by identify invalid embedded tokens or capabilities, similar to the capability approaches in IP networks [3], [4], [17], [18], [22], [27].

Recently, encryption is used for data protection in ICN [6], [9], [19], [21], [24], [25], [27]. For instance, DiBenedetto *et al.* [9] leverage onion routing to ensure anonymity of NDN. Nabeel *et al.* apply homomorphic cryptography to support data lookup over encrypted packets in publish-subscribe networks [21]. CSEA is orthogonal to these application layer encryption approaches, and focuses on distributed access control in the network layer.

Network-layer trust in NDN [12] and unpredictable names [2], [7] are mostly related to our proposed CSEA. The network-layer trust approach [12] uses one hash function in each node to verify CPs (more precisely, it verifies CPs' public keys) and content, so as to build trust among CPs, routers, and users. Similar to CSEA, it incurs small computation overhead to verify content. However, it does not consider DoS attacks and content leakage attacks that are addressed in CSEA. Our current CSEA design is similar to the approaches with unpredictable names [2], [7] that aim to defend against the content poisoning attacks. To completely prevent the attacks that sniff and replay interest packets, we can extend the scheme and encrypt the tokens in interest packets, at the expense of incurring higher packet delivery overheads.

IX. Conclusion

In this paper, we propose CSEA to enforce security policies in NDN such that it can throttle different attacks in NDN, i.e., content poisoning attacks, DoS attacks, and content leakage attacks. Specially, we leverage lightweight hash algorithms to implement a capability system within CSEA. We implement the CSEA prototype upon the CCNx platform, and demonstrate the benefits of CSEA with testbed and Planetlab experiments. The experimental results show that CSEA introduces negligible overhead in retrieving data packets in NDN.

References

[1] *The Content Centric Networking (CCNx) Project*. [Online]. Available: http://www.ccnx.org/

[2] G. Acs, M. Conti, P. Gasti, C. Ghali, and G. Tsudik, "Cache privacy in named-data networking," in *Proc. ICDCS*, Jul. 2013, pp. 41–51.

[3] T. Anderson, T. Roscoe, and D. Wetherall, "Preventing Internet denial-of-service with capabilities," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 1, pp. 39–44, 2004.

[4] K. Argyraki and D. Cheriton, "Network capabilities: The good, the bad and the ugly," in *Proc. Hotnets*, 2005, p. 140.

[5] M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proc. CCS*, 1993, pp. 62–73.

[6] H. Ben Abraham *et al.*, "Tutorial: Security and synchronization in named data networking (NDN)," in *Proc. ICN*, 2015, pp. 3–6.

[7] A. Compagno, M. Conti, P. Gasti, L. V. Mancini, and G. Tsudik, "Violating consumer anonymity: Geo-locating nodes in named data networking," in *Proc. ACNS*, 2015, pp. 243–262.

[8] A. Compagno, M. Conti, P. Gasti, and G. Tsudik, "Poseidon: Mitigating interest flooding DDoS attacks in named data networking," in *Proc. IEEE Conf. Local Comput. Netw.*, 2013, pp. 630–638.

[9] S. DiBenedetto, P. Gasti, G. Tsudik, and E. Uzun, "ANDaNA: Anonymous named data networking application," in *Proc. NDSS*, 2012.

[10] N. Fotiou, G. F. Marias, and G. C. Polyzos, "Access control enforcement delegation for information-centric networking architectures," in *Proc. ICN*, 2012, pp. 85–90.

[11] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang, "DoS and DDoS in named-data networking," in *Proc. ICCCN*, 2013.

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

[13] L. Gong, "A secure identity-based capability system," in *Proc. IEEE Symp. Secur. Privacy*, 1989, pp. 56–63.

[14] V. Jacobson *et al.*, "Networking named content," *Commun. ACM*, vol. 55, no. 1, pp. 117–124, Jan. 2012.

[15] H. M. Levy, *Capability-Based Computer Systems*. Newton, MA, USA: Butterworth, 1984.

[16] Q. Li, X. Zhang, Q. Zheng, R. Sandhu, and X. Fu, "LIVE: Lightweight integrity verification and content access control for named data networking," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 2, pp. 308–320, Feb. 2015.

[17] X. Liu, X. Yang, and Y. Lu, "To filter or to authorize: Network-layer dos defense against multimillion-node botnets," in *Proc. ACM SIGCOMM*, 2008, pp. 195–206.

[18] X. Liu, X. Yang, and Y. Xia, "NetFence: Preventing Internet denial of service from inside out," in *Proc. ACM SIGCOMM*, 2010, pp. 255–266.

[19] A. K. Maji and S. Bagchi, "*v*-CAPS: A confidentiality and anonymity preserving routing protocol for content-based publish-subscribe networks," in *Proc. SecureComm*, 2011, pp. 281–302.

[20] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Proc. CRYPTO*, 1987, pp. 369–378.

[21] M. Nabeel, N. Shang, and E. Bertino, "Efficient privacy preserving content based publish subscribe systems," in *Proc. SACMAT*, 2012, pp. 133–144.

[22] B. Parno *et al.*, "Portcullis: Protecting connection setup from denial-of-capability attacks," in *Proc. ACM SIGCOMM*, 2007, pp. 289–300.

[23] J. S. Shapiro, J. M. Smith, and D. J. Farber, "EROS: A fast capability system," in *Proc. SOSP*, 1999, pp. 170–185.

[24] M. Srivatsa, L. Liu, and A. Iyengar, "Eventguard: A system architecture for securing publish-subscribe networks," *ACM Trans. Comput. Syst.*, vol. 29, no. 4, pp. 10:1–10:40, 2011.

[25] M. A. Tariq, B. Koldehofe, and K. Rothermel, "Securing broker-less publish/subscribe systems using identity-based encryption," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 518–528, Feb. 2014.

[26] C. A. Wood and E. Uzun, "Flexible end-to-end content security in CCN," in *Proc. IEEE CCNC*, Jan. 2014, pp. 858–865.

[27] X. Yang, D. Wetherall, and T. Anderson, "TVA: A DoS-limiting network architecture," *IEEE/ACM Trans. Netw.*, vol. 16, no. 6, pp. 1267–1280, Dec. 2008.

[28] C. Yi *et al.*, "A case for stateful forwarding plane," *Comput. Commun.*, vol. 36, no. 7, pp. 779–791, Apr. 2013.

[29] C. Yi, A. Afanasyev, L. Wang, B. Zhang, and L. Zhang, "Adaptive forwarding in named data networking," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 3, pp. 62–67, 2012.

[30] Y. Yu *et al.*, "Schematizing trust in named data networking," in *Proc. 2nd ACM Conf. Inf.-Centric Netw.*, 2015, pp. 177–186.

[31] K. Zhang, "Efficient protocols for signing routing messages," in *Proc. NDSS*, 1998.

Qi Li received the Ph.D. degree from Tsinghua University. He was with ETH Zurich, the University of Texas at San Antonio, The Chinese University of Hong Kong, the Chinese Academy of Sciences, and Intel. He is currently an Associate Professor with the Graduate School at Shenzhen, Tsinghua University. His research interests are in network and system security, particularly in Internet and cloud security, mobile security, and big data security. He is currently an editorial board member of the IEEE Transactions on Dependable and Secure Computing.

Patrick P. C. Lee received the B.Eng. degree (Hons.) in information engineering from the Chinese University of Hong Kong in 2001, the M.Phil. degree in computer science and engineering from the Chinese University of Hong Kong in 2003, and the Ph.D. degree in computer science from Columbia University in 2008. He is currently an Associate Professor with the Department of Computer Science and Engineering, Chinese University of Hong Kong. His research interests are in various applied/systems topics including storage systems, distributed systems and networks, operating systems, dependability, and security.

Peng Zhang received the Ph.D. degree in computer science from Tsinghua University in 2013. He is currently an Associate Professor with the Department of Computer Science and Technology, Xi'an Jiaotong University, China. He has been a Visiting Student with the The Chinese University of Hong Kong and Yale University. His research interests include network security and privacy, software-defined networks, and information-centric networks.

Purui Su received the Ph.D. degree from the Chinese Academy of Science in 2007. He is currently a Professor with the Institute of Software, Chinese Academy of Sciences. His current research interests include computer security, program security, network security, and mobile security.

Liang He received the Ph.D. degree from the Chinese Academy of Science in 2012. He is currently an Assistant Professor with the Institute of Software, Chinese Academy of Sciences. His current research interests include computer security, program security, and network security.

Kui Ren received the Ph.D. degree from the Worcester Polytechnic Institute. He is currently a Professor of computer science and engineering and the Director of the UbiSeC Laboratory with University at Buffalo, The State University of New York. His current research interests span cloud and outsourcing security, wireless and wearable system security, and humancentered computing. He currently serves as an Associate Editor of the IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, the IEEE TRANSACTIONS ON MOBILE COMPUTING, the IEEE TRANSACTIONS ON SMART GRID, the IEEE WIRELESS COMMUNICATIONS, and the IEEE INTERNET OF THINGS JOURNAL.