Hindawi
Security and Communication Networks
Volume 2018, Article ID 5429798, 17 pages
https://doi.org/10.1155/2018/5429798

Research Article

TLDA: An Efficient Two-Layered Data Authentication Mechanism for Content-Centric Networking

Seog Chung Seo[1] **and Taek-Young Youn**[2]

[1]*The Affiliated Institute of ETRI, Daejeon, Republic of Korea*
[2]*ETRI, Daejeon, Republic of Korea*

Correspondence should be addressed to Taek-Young Youn; t.y.youn@gmail.com

Received 8 December 2017; Revised 8 May 2018; Accepted 24 May 2018; Published 4 July 2018

Academic Editor: Huaizhi Li

Content-Centric Networking (CCN) is a new networking paradigm for the future Internet, which shifts the communication paradigm from host-centric to data-centric. In CCN, contents are routed by their unique names and they are stored in network nodes by units of segment during transmission for future usage. Since contents are stored in network nodes in a distributed manner, security is built into CCN data packets by embedding a public key signature to enable any content requesters to verify authenticity and integrity of contents. However, the use of public key signatures for authenticating CCN data packets incurs significant overhead regarding computation and communication, which limits universal utilization of CCN. Furthermore, this can lead to a new kind of DDoS attacks. Even though CCN adopts an aggregate signature method based on Merkle Hash Tree (MHT) in its reference implementation, it still incurs large amount of overhead. This paper presents TLDA, an efficient Two-Layered Data Authentication mechanism, which can considerably reduce overhead of computation and communication for authenticating data segments in CCN. For efficiency of computation and communication, TLDA newly introduces the concept of authentication Meta part consisting of data segments' hash values. To a great extent TLDA not only reduces the computation and communication overhead compared with CCN's basic authentication method, but also provides robustness against transmission loss and out-of-order transmission. We have implemented TLDA and demonstrated that it provides 74.3% improved throughput and 36.557% reduced communication overhead compared to those of the original CCNx library developed by PARC when transmitting a 128Mbyte content in units of 1Kbyte segment with RSA-2048 and SHA-256 as its signature algorithm and hash algorithm, respectively.

1. Introduction

Content-Centric Networking (CCN) has recently been proposed as one of different types of Information-Centric Networking (ICN) [1–5] to solve the problem of heavy traffic increases and the existing IP-based Internet's problems [1]. In CCN, contents are routed by their own content names rather than IP-addresses and data segments in the contents are stored in network nodes' storage (so-called in-network caching) during transmission for handling future requests rapidly. CCN's name-based routing and in-network caching enable multipath content forwarding and optimization of bandwidth usage, which resolves the current Internet's data traffic problems and DDoS (Distributed Denial-of-Service) security attacks. Since CCN allows contents to be stored in network nodes by units of segment, it employs public key signature as the built-in data and source authentication mechanism. While use of digital signature certainly provides built-in security such as data integrity and authentication to CCN data segments, it imposes significant computational overhead on the network nodes.

To address heavy overhead from public key signature-based data segment authentication in CCN, several researches have been conducted [1, 6–14]. These can be divided into five categories: Merkle Hash Tree (MHT) based signing approaches [1, 6–8], HMAC-based methods [9, 10], self-verifying name-based methods [11, 12], hash chain-based method [13], and multitoken-based method [14]. PARC (Palo Alto Research Center) implemented MHT-based aggregate signing mechanism into CCNx, which is the reference

implementation of CCN protocol [1, 4, 7, 8]. Even though the use of MHT in CCNx could reduce the amount of signing and verification, it still incurs heavy amount of overhead in computation and communication compared with the case without segment authentication. HMAC-based methods [9, 10] could handle the overhead of digital signature-based packet authentication. However, they require bootstrap system setup and secure distribution of HMAC keys. Self-verifying name-based methods [11, 12] are capable of efficient data authentication. However, they rely on a certain naming architecture and the authenticity of self-verifying name needs to be provided through additional notary entity [11] or periodical broadcasting [12]. Hash chain-based mechanism was proposed for efficient data segment authentication in CCN [13]. Even though it is efficient with regard to computation and communication, it might be vulnerable to transmission error due to inherent property of hash chain. Noteworthily, in 2015, multitoken-based method was proposed for efficient data integrity checking and access control in CCN nodes [14]. Even if transmission delay and communication overhead from the method are acceptable, it requires that a content publisher has complete identity information of authorized CCN nodes along with their public keys.

In this paper, we present TLDA, Two-Layered Data Authentication, mechanism for CCN, to provide an efficient data segment authentication mechanism in CCN. For efficiency of data segment authentication, TLDA newly introduces the concept of authentication Meta part. Since the authentication Meta part holds the information essential for verifying data segments, TLDA applies MHT to segments in authentication Meta part to provide robustness against transmission errors. The process of generating authentication information in TLDA mainly consists of two phases: Data part encoding phase and Meta part encoding phase. In other words, at Data part encoding phase, TLDA firstly divides a content into fixed-size segments and converts them into CCN data segments and then generates their hash values. At Meta part encoding phase, TLDA constructs Meta part and every segment in Meta part consists of hash values previously generated from hash value generation phase. Since Meta part is a crucial part for conveying authentication information necessary for validating data segments in the proposed mechanism, TLDA applies MHT-based signing to segments in Meta part. Thus segments in Meta part can be independently verified without depending on other packets in Meta part. Once a content requester verifies a segment in Meta part, he/she can begin to verify data segments just with hash value comparison by using the hash values contained in the Meta part segment. Through the two-layered authentication mechanism with MHT-based Meta part segment authentication and data segment authentication using hash value comparison, TLDA attains both efficient data segment authentication and robustness against transmission errors. Note that TLDA is particularly suitable for processing constant contents such as VOD, installation files, and document files since it is easy to construct authentication Meta part from previously generated data segments. The contributions in this paper are threefold:

(i) We propose TLDA, which is an efficient Two-Layered Data Authentication mechanism for CCN. For efficiency of authentication, TLDA constructs authentication Meta part from data segments and applies MHT-based signing to segments in Meta part. Thus segments in Meta part can be verified independently while providing robustness against transmission errors and data segments can be authenticated with efficient hash value comparison.

(ii) We prototype TLDA and implement it in CCNx library. Our experimental results show that it incurs acceptable amount of overhead. In particular, when transmitting a 128Mbyte content in units of 1Kbyte segment, TLDA, which uses RSA-2048 [15] and SHA-256 [16] as its signature and hash algorithm, incurs only 3.423% of time delay and requires 7.877% of communication overhead compared to the case that transmits the same content without data segment authentication. At this time, TLDA provides that about 74.3% improved throughput and 36.557% reduced communication overhead compared to those of the original CCNx library using MHT-based segment authentication mechanism.

(iii) We analyze the performance of TLDA and compare it with CCNx library's original MHT-based segment authentication mechanism on various test parameters such as segment size and applied crypto algorithms in terms of computation and communication overhead.

This paper is organized as follows. We briefly review the related works including CCN and existing research results for achieving CCN data authentication and MHT-based authentication in Section 2. Section 3 describes details of TLDA at publisher and requester sides. Section 4 analyzes TLDA as to security and theoretical computation overhead. Section 5 compares TLDA with the segment authentication mechanism in CCNx in terms of computation and communication overhead and Section 6 concludes this paper.

2. Related Work

In this section, we review CCN design, previously presented results of authenticating CCN data segments, and MHT-based authentication.

2.1. Content-Centric Networking (CCN) and Content Authentication in CCN. Content-Centric Networking (CCN) has been proposed as a new networking architecture delivering data packets by content names rather than host addresses [1, 4]. CCN is characterized by name-based routing, in-network caching, and built-in data packet security. In CCN, two types of packet are defined: Interest packet and Data packet [1]. Interest packet is a content request packet and it contains the name of a requested content. Data packet is a response packet corresponding to Interest packet and it consists of the content name, signature, signed info, and data field. Signature field contains the authentication information on the data packet and the data field contains the real data piece. The signed info field contains public key itself or information enabling retrieval of the public key necessary for verifying the signature. Since CCN allows the content to be stored by units of data packet (segment) in network nodes in order to attain bandwidth optimization through in-network caching,

security mechanisms for data integrity and authenticity are required. CCN provides a public key signature-based data packet authentication method as a built-in integrity and authentication mechanism. Although using digital signature certainly fulfills data integrity and authentication for CCN data packets, it causes significant overhead from signature generation and verification on network nodes including publishers and requesters, which limits widespread CCN utilization and may lead to a new kind of DDoS attacks [17].

Recently several researches have been conducted to deal with the problem of heavy overhead from public key signature-based data packet authentication in CCN [1, 4, 7–14]. PARC implemented MHT-based aggregate signing mechanism into CCNx, which is the reference implementation of CCN protocol [1, 4, 7, 8]. However, CCNx still suffers from heavy overhead in computation and communication. For example, MHT-based segment authentication mechanism equipped in CCNx incurs 44.592% of time delay in accessing contents and requires 44.434% of communication overhead with RSA-2048 and MHT of size 32. In 2013, Jeff Burke et al. utilized CCN to control lighting in Building Automation System, where they used Interest messages as lighting control messages [9, 10]. Since Interest packets were used for controlling sensitive building equipment, they introduced HMAC-based authenticated Interest to improve low performance of public key-based authenticated Interest. Although their mechanism could significantly save the cost for authenticating Interest packets compared with the existing mechanism using RSA-1024, it requires system setup and secure distribution of HMAC keys, which make their mechanism limited to small-scale network environments rather than wide-ranging environments. There have been some works introducing the use of self-verifying content name to authenticate data packet with minimal use of digital signature [11, 12]. In self-verifying naming methods, the hash value of the data packet becomes its content name. If the content name is authentic, the authenticity of the data packet can be validated by comparing the received data packet's hash value with its name. In 2012, Mark Baugher et al. proposed a CCN data packet authentication method using self-verifying content name and manifest file containing the list of self-verifying names for data packets in a content [11]. Their mechanism added a notary entity for signing manifest files to make the self-verifying names authentic. A content requester first needs to receive the manifest file of a content in order to access the content. When the requester receives it, he/she validates the file by verifying the signature signed by the notary entity. Then the requester requests data packets for the content by sending Interest packets including self-verifying names obtained from the manifest file. In 2014, Ilya Moiseenko introduced a similar idea of using self-verifying name-based data packet authentication [12]. Rather than using a notary entity, his method periodically distributes manifest file to requesters. Even though self-verifying name-based methods can reduce the number of signature generation and verification, they depend on a certain naming architecture and the works in [11, 12] did not provide the implementation results. In 2015, Qi Li et al. proposed LIVE (Lightweight Integrity Verification architecture), which provides an efficient data segment integrity mechanism and allows a content publisher to control access to contents stored in remote CCN network nodes [14]. In LIVE, a content publisher provides different kinds of tokens to network nodes according to the access control policy and only nodes that receive the authorized token can verify the content and store it in their Content Store. Even though LIVE incurs average 10% of time delay only in accessing contents and around 15% of commutation overhead in accessing to a content with RSA-1024 and SHA-1, it requires that a content publisher have complete identity information on authorized CCN nodes along with their public keys, which may not be achievable all the time. Moreover, it is expected that LIVE incurs more than 10% delay and requires more than 15% communication overhead if it makes use of RSA-2048 and SHA-256 as its signature algorithm and hash algorithm. In 2015, Tamper Refaei et al. proposed a HC- (Hash Chain-) based data authentication method in CCN [13]. They consider backward and forward hash chains to provide data and source authentication. Although their method provides efficient performance in computation and communication, it can be vulnerable to transmission errors such as transmission loss and out-of-order transmission due to the inherent property of hash chain.

2.2. Merkle Hash Tree- (MHT-) Based Authentication and Its Use in CCNx. To reduce the overhead of signature signing and verification processes, Merkle Hash Tree (MHT) [18, 19] has been widely used [1, 7, 8, 20–23]. A sender buffers k data blocks and signs only the root hash node of the binary hash tree about the blocks. The sender sends a packet including a data block, its witness (the hash tree path to the root node), and the root signature to receivers. When the receiver receives the packet, he/she can verify its validity by computing the root hash using the received data block and the witness and comparing the computed root hash value with the root hash value from the root signature verification. The receiver can cache the verified hash values at the authenticated hash table (AHT) for quick verification of the other data blocks which belong to the same hash tree, which is called implicit authentication. In MHT-based authentication, each data block can be verified individually based on its witness information and the root signature without requiring any information in other blocks. However, when using MHT-based authentication method, the size of tree should be carefully determined because the more the size of tree in MHT for improving the performance of signing and verification, the more the communication overhead. CCNx library from PARC makes use of MHT as its basic segment authentication mechanism [1, 6–8]. For example, CCNx library applies MHT-based signing to all segments in the content. When a content publisher receives Interest packets requesting segments of the content, he/she reads k (the size of leaf nodes in MHT) segments from the content file at a time and constructs an MHT for the segments and then finally generates a root signature. CCNx library repeats this MHT-based signing process by units of k segments until it finishes the process for generating segment authentication information while responding to Interest packets from the content requester, simultaneously. Even though CCNx library

could achieve individual segment authentication, the naive application of MHT-signing to all segments incurs heavy amount of computation and communication overhead. The detailed comparison between TLDA and CCNx library will be given in Section 5.

3. Proposed Authentication Mechanism

In this paper, we propose TLDA, which provides efficient data segment authentication in CCN. TLDA not only provides immediate data segment authentication, but also achieves much enhanced performance with respect to computation and communication compared to CCNx.

3.1. Overview and Notations. TLDA aims to achieve the following design goals:

(i) Lightweight data segment authentication to prevent CCN nodes from accessing "corrupted" or "fake" contents.

(ii) Providing not only efficiency with respect to computation and communication but also robustness against transmission errors.

This section gives a brief overview and introduces notations, and following sections present the detailed design of the proposed mechanism. TLDA is composed of segment authentication information generation mechanism at content publisher (sender) side and segment verification mechanism at content requester (receiver) side. Differently from the CCNx library, which applies MHT-based signing to all segments in a content, TLDA newly introduces authentication Meta part and applies MHT-based signing only to segments in Meta part. In TLDA, only a small number of Meta part segments are able to convey entire information which is required to authenticate all data segments. That is, with a small number of Meta part segments, a content requester can verify the validity of data segments immediately by using the authentication information embedded in the Meta part segments.

Authentication information generation process in TLDA consists of two steps: Data part encoding and Meta part encoding. The main concerns of Data part encoding are segmentation, CCN data segment formatting, and hash value generation of data segments. As same as CCN, TLDA divides an original content into fixed-size data segments and converts them to (CCN) data segments. The generated data segments have their own unique name and the set of data segments constitute Data part in TLDA. Then TLDA generates hash values of data segments in Data part. At this time, TLDA uses cryptographic hash functions such as SHA-256. The main concerns of Meta part encoding are Meta part formatting and Meta part signing. At first, TLDA generates Meta part segments by using previously generated hash values and formats them as CCN segments. We assume that a segment in Meta part consists of w data segments' hash values (since the basic segment size in CCNx library is 4Kbytes, if SHA-256 is used for cryptographic hash function, w will be 128). After Meta part formatting, each Meta part segment contains its own unique name and w hash values in its payload. Since segments in Meta part convey crucial information for authenticating data segments, TLDA applies MHT-based signing to them. With application of MHT-based signing, each segment in Meta part can be verified independently, which provides robustness against transmission errors. Because of the introduction of authentication Meta part, TLDA can provide efficient data segment authentication with almost $(1/w)$ reduced overhead of computation and communication compared to CCNx library proposed by PARC.

In CCN, content requesters call for content delivery by sending Interest packets containing the name of the content. In TLDA, first of all, content requesters send Interest packets for Meta part segments since segments in Meta part are prerequisite for authenticating data segments. When the received segment in Meta part is proven to be valid, the hash values included in the segment are stored in the AHT (Authenticated Hash Table) to be used when verifying Data part segments. In other words, w hash values in a Meta part segment can be used to verify w segments in Data part.

Table 1 shows the notations for describing our proposed mechanism. We assume that the original content C consists of n segments of size $|s|$. w is the number of hash values which can be stored in a segment of size $|s|$. If we assume that the size of hash value is 32bytes and the size of a segment is 4Kbytes, w is 128 (in CCNx implementation, the basic segment size is 4Kbytes). In TLDA, the original content C is converted into Data part D, where segments have their own names such as *MName* (in CCN, each segment has its name because it is routed based on its name). In this sense, s_i^*, which is a segment in Data part, contains its unique name $DName_i$ and original data payload s_i. A Meta part segment m_i^* includes w hash values $h_{iw} \parallel \ldots \parallel h_{(i+1)w-1}$, where h_{iw} is the hash value of data part segment s_{iw}^*. Similar to Data part segments, Meta part segments have their own unique name *MName*. After the construction of Meta part, TLDA applies MHT-based signing to segments in Meta part. As a result, a segment $m_i^\dagger$ in signed Meta part contains an original Meta part segment m_i^*, root signature Sig_R, and witness information wit_i, which is a set of hash values used to compute the root hash value h_R. *PRK*, *PUK* are a signing key (private key) and verification key (public key), respectively. The final authenticated content is C^* and it is composed of signed Meta part and Data part like $(M^* \parallel D)$.

3.2. Process for Generating Authentication Information of Segments. In this section, we describe the detail of authenticated segment generation process in TLDA. Figure 1 shows overall process for authenticated segment generation in TLDA. Process for generating authenticated segments consists of two phases: Data part encoding and Meta part encoding. Data part encoding consists of segmentation, formatting, and segments' hash value generation. Meta part encoding is composed of formatting and MHT-based signing. Because of the deterministic property of hash function, hash value generation in Data part encoding can be processed in offline. On the other hand, MHT-based Meta part signing needs to be conducted in online considering the tradeoff between computational performance improvement and increase of

Table 1: List of notations.

Notation	Meaning
PRK, PUK	private key for signing, public key for verification
$Hash$, $Sign$, $Verify$	Hash operation, RSA Sign, and Verify
h_R, hn, wit, Sig_R	root hash, hash node, witness(authentication path) information, and root signature in MHT
C	original content
$\|C\|$, $\|H\|$, $\|s\|$	size of content, size of hash, and size of segment
n	the number of segments in C
w	the number of hash values which can be stored in a segment of $\|s\|$
s, s_i	original data segment, i-th data segment in original content where $0 \le i \le n-1$
D	Data part
$DName$, h	content name field and hash value of segment in Data part
s_i^*	i-th segment in Data part D, $s_i^* = DName_i \parallel s_i$ where $0 \le i \le n-1$
h_i	hash value of i-th segment in Data part D, $h_i = Hash(s_i^*)$ where $0 \le i \le n-1$
M	Meta part
N	$N = n/w$, the number of Meta part segments (Assume n is multiple of w)
$MName$, H	content name field and hash value of segment in Meta part
m_i^*	i-th segment of Meta part, $m_i^* = MName_i \parallel m_i$ where $m_i = h_{iw} \parallel \ldots \parallel h_{(i+1)w-1}$ and $0 \le i \le N-1$
H_i	$H_i = Hash(m_i^*)$ where $0 \le i \le N-1$
M^*	signed Meta part
$m_i^\dagger$	i-th segment in signed Meta part M^*, $m_i^\dagger = m_i^* \parallel wit_i \parallel Sig_R$ where wit_i is the authentication path(sibling path)
C^*	Encoded content = (signed Meta part $\parallel$ Data part) = $(M^* \parallel D)$

communication overhead. Note that a content name field and a signed info field are generated and included in each segment of Data part and Meta part through formatting process. Then, when computing hash value of a data segment, the concatenation of the content name field, the signed info field, and the data field is hashed. Details in both Data part encoding and Meta part encoding will be described in the following subsections.

3.2.1. Data Part Encoding. Data part encoding process operates segmentation, formatting, and hash value generation, sequentially. Figure 2 shows the process of Data part encoding. At the segmentation step, the original content C is divided into n segments of fixed size $|s|$ (this process can be conducted by iteratively reading a content file as unit of $|s|$ and generating CCN data segment object). Then segments from the segmentation are converted into CCN data segments through the formatting step. Specifically, a unique name $DName$ for identifying each segment is assigned to each segment's content name field and original segment s is stored in data field of the formatted data segment (for simplicity, we omit the signed info and signature fields). The set of generated CCN data segments is defined as Data part. When computing a hash value of a data segment, the concatenation of content name field, data field, and signed info field is hashed like $Hash(DName_i \parallel s_i)$, where $0 \le i \le (n-1)$.

Algorithm 1 generates Data part segments from Step 1 to 13. Step 1–5 divides a content into n segments of size $|s|$. Step 6-9 formats data segments in Data part by assigning unique name to each segment. Then step 10-13 computes segments' hash values. Since TLDA requires hash value computation of data segments, it is suitable for authenticating constant contents such as VOD, installation files, and document files.

3.2.2. Meta Part Encoding. Meta part encoding process consists of Meta part segment formatting and MHT-based signing. After finishing Data part encoding, TLDA constructs Meta part, where its segments are composed of hash values from Data part encoding (refer to Figure 3). Step 14-18 in Algorithm 1 formats Meta part segments by assigning a unique name $MName$ to them and storing hash values generated from Data part encoding in their data field. Each Meta part segment contains w hash values in its data field, which are used to verify the authenticity of Data part segments. For example, i-th segment m_i^* in Meta part contains w hash values $(h_{iw}||\ldots||h_{(i+1)w-1})$, where $0 \le i \le N-1$. Since each segment in Meta part contains hash values which are used to verify the authenticity of Data part segments, it is important to deliver it in a reliable condition. Thus, TLDA applies MHT-based signing to Meta part segments to make them resistant against transmission errors. Although MHT-based signing mechanism increases communication overhead, it makes

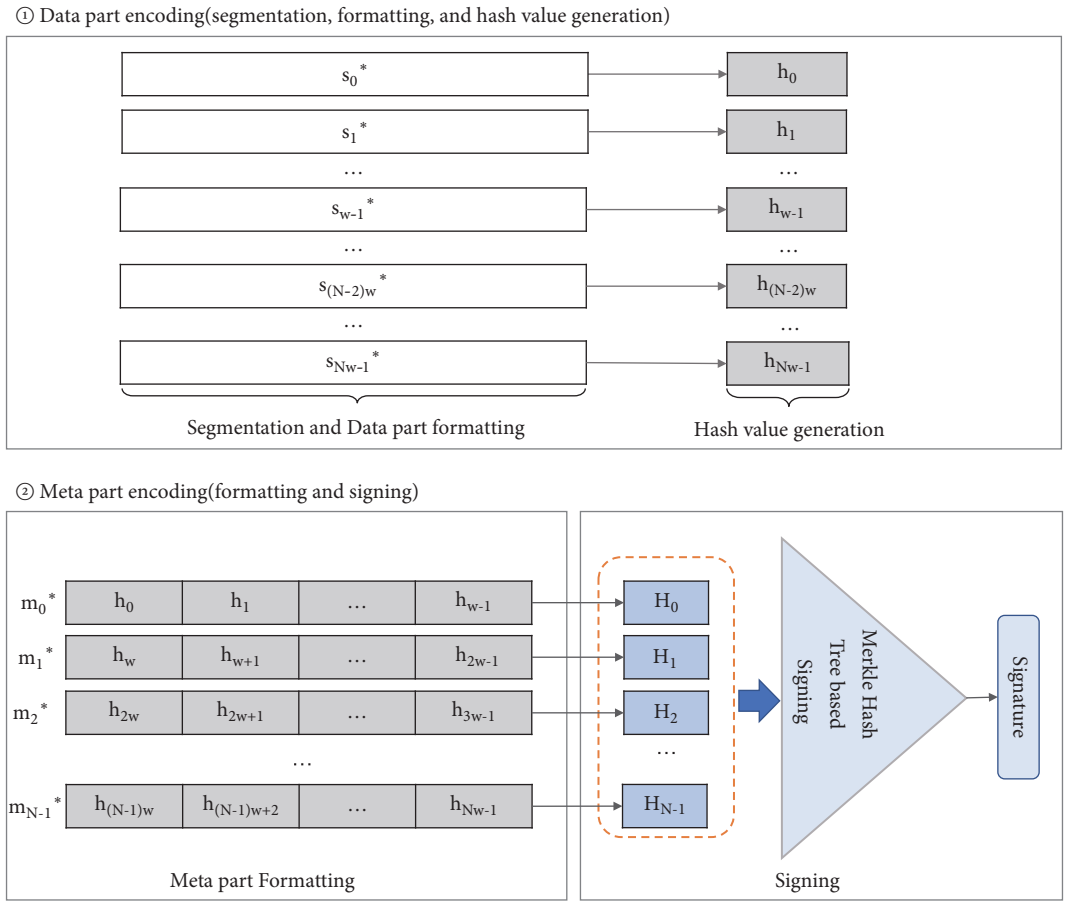

Figure 1: Process for generating authentication information in TLDA.

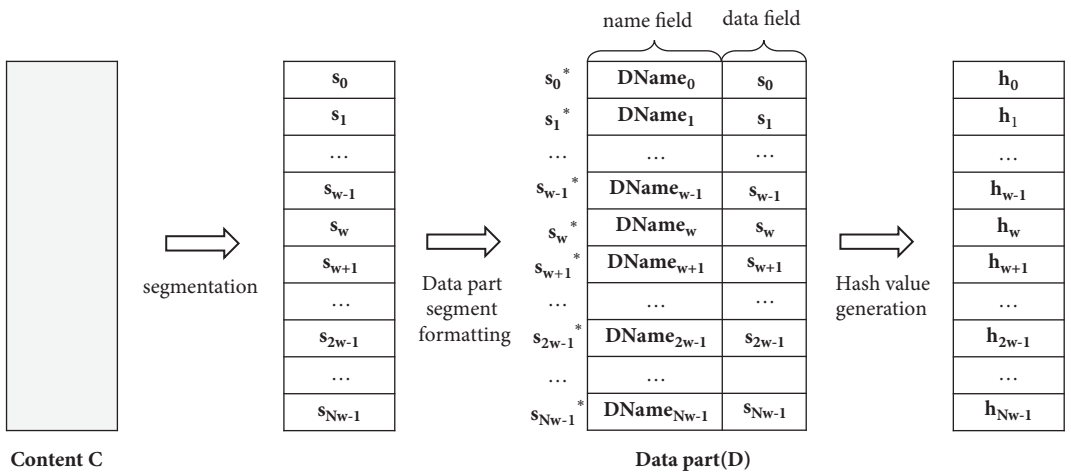

Figure 2: Data part encoding process. (For simple description, we omit signed info field in each segment in this figure.)

each segment in Meta part verified independently without depending on any other segments. Step 19–26 in Algorithm 1 applies MHT-based signing to Meta part segments. Firstly, at step 20-22, the leaf hash values $(H_0, H_1, H_2, \ldots, H_{N-1})$ are computed, and then MHT-based signing is applied to them by calling Algorithm 2. Algorithm 2 makes the number of leaf nodes 2's power by padding $(2^L - k)$ as zero value in Step 2. Then it constructs binary Merkle Hash Tree from leaf nodes until a root node is computed in Step 4–13. Algorithm 2 utilizes an array type of memory in order to store the constructed MHT. Thus, the parent nodes are stored in the increased index of *hn* and at last, the root hash value

Require: Content C, segment size $|s|$, private PRK.
Ensure: Authenticated content composed of encoded Meta part and encoded Data part.
1. /* Data part encoding: segmentation */
2. $n = |C|/|s|$
3. **for** i from 0 downto $n - 1$ **do**
4. $\quad s_i \longleftarrow C[is, ..., (i+1)s - 1]$
5. **end for**
6. /* Data part encoding: formatting */
7. **for** i from 0 downto $n - 1$ **do**
8. $\quad s_i^* \longleftarrow (DName_i \parallel s_i)$
9. **end for**
10. /* Data part encoding: hash generation */
11. **for** i from 0 downto $n - 1$ **do**
12. $\quad h_i \longleftarrow Hash(DName_i \parallel s_i^*)$
13. **end for**
14. /* Meta part encoding: formatting */
15. $N = n/w$
16. **for** i from 0 downto $N - 1$ **do**
17. $\quad m_i^* \longleftarrow MName_i || (h_{iw} \parallel \ldots \parallel h_{(i+1)w-1})$
18. **end for**
19. /* Meta part encoding: MHT-based Signing */
20. **for** i from 0 downto $N - 1$ **do**
21. $\quad H_i \longleftarrow Hash(m_i^*)$
22. **end for**
23. Call $MHTSign(H_0, \ldots, H_{N-1}, PRK)$
24. **for** i from 0 downto $N - 1$ **do**
25. $\quad m_i^\dagger \longleftarrow m_i^* \parallel wit_i \parallel Sig_R$
26. **end for**
Return: Authenticated content ($C^* = M^* \parallel D$)

Algorithm 1: Authenticated content generation.

Require: k hashes $(hn_0, \ldots, hn_{(k-1)})$, private PRK.
Ensure: Merkle hash tree, root signature Sig_R.
1. Find smallest L such that $k \leq 2^L$
2. Pad $(2^L - k)$ leaf nodes $(hn_k, \ldots, hn_{(2^L-1)})$ with zero value
3. $Start \longleftarrow 0$, $End \longleftarrow 2^L - 1$
4. **for** i from 0 downto $L - 1$ **do**
5. $\quad PIndex \longleftarrow End + 1$
6. $\quad$ **for** j from $Start$ downto End by 2 **do**
7. $\quad\quad Left \longleftarrow hn_j$, $Right \longleftarrow hn_{(j+1)}$
8. $\quad\quad hn_{PIndex} \longleftarrow Hash(Left \parallel Right)$
9. $\quad\quad PIndex \longleftarrow PIndex + 1$
10. $\quad$ **end for**
11. $\quad Start \longleftarrow Start + (2^{(L-i)})$
12. $\quad End \longleftarrow End + (2^{(L-i-1)})$
13. **end for**
14. $h_R \longleftarrow hn_{(2^{L+1}-2)}$
15. $Sig_R \longleftarrow Sign(h_R)$
Return: Merkle hash tree $(hn_0, \ldots, hn_{(2^{(L+1)}-2)})$, root signatgure Sig_R

Algorithm 2: Merkle Hash Tree-based signing ($MHTSign()$).

is stored in $hn_{(2^{(L+1)}-2)}$. After finishing the construction of MHT, Algorithm 2 generates a root signature of the MHT by signing the MHT's root hash value in Step 15. Figure 4 shows the application of MHT-based signing to Meta part segments. After applying MHT-based signing, each segment in encoded Meta part has a root signature Sig_R and its witness information, which is used to compute the root hash value h_R in the MHT at the requester side. Step 24–26 in Algorithm 1 encodes segments in Meta part by storing the witness information wit_i and signature Sig_R generated from Step 23 in its signature field.

Figure 5 shows the example of finally generated authenticated content through Data part encoding and Meta part encoding in TLDA. Since the proposed mechanism uses hierarchical naming in the example, the format of the segment name becomes contentname/Meta|Data/seg#. We assume that the content name is contentfile. Through the proposed encoding process, two types of encoded parts are generated from a content, and segments in each encoded part are identified with their unique name like contentfile/Meta/seg_i or contentfile/Data/seg_i. In Figure 5, all segments in Meta part contain their witness information (wit) and root signature (Sig_R) in the MHT and convey w hash values necessary for authenticating Data part segments in their data field. Note that there is no signature field in a Data part segment in TLDA. This is because in TLDA the use of signature field in a data segment is superseded by the introduction of authentication Meta part.

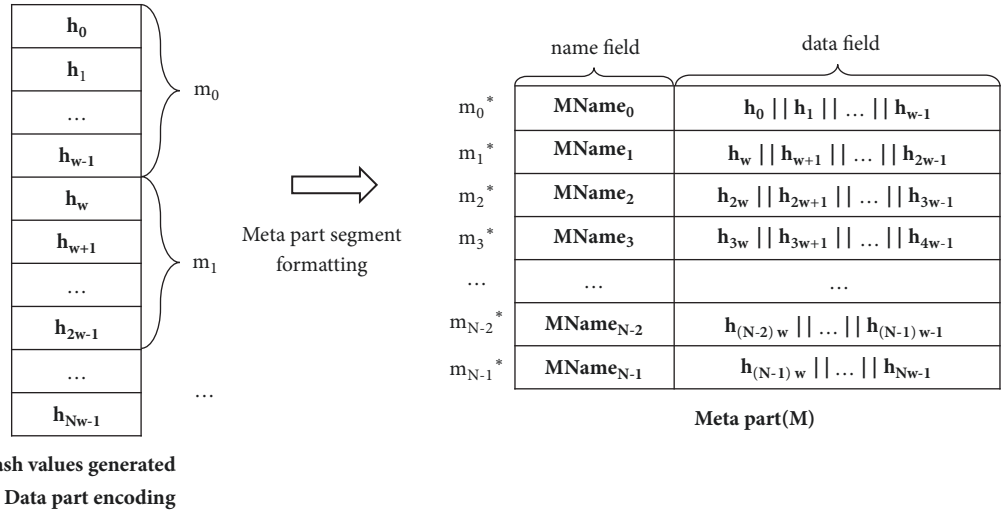

Figure 3: Meta part encoding: Meta part segment formatting. (For simple description, we omit signed info field in each segment in this figure.)

3.3. Segments Verification. Figure 6 shows the overall segment verification process in TLDA. The verification process consists of two parts: Meta part segment verification and Data part segment verification. The following subsections describe the detail of each process.

3.3.1. Verifying Segments in Meta Part. In CCN, content requesters call for content delivery by sending Interest packets containing the name of the content. In the proposed mechanism, first of all, content publisher sends signed Meta part M^* because segments in M^* hold hash values which are necessary for verifying segments in Data part. When requesters receive a segment in Meta part, they can authenticate it with either MHT-based signature verification or MHT's implicit authentication. When the received segment in Meta part is proven to be valid, the hash values included in the segment are stored in the AHT (Authenticated Hash Table) for verifying segments in Data part. For example, when $m_i^\dagger$ is received, requesters firstly search the AHT for checking whether there exist hash values related to the segment or not. If one of the segments in the same MHT has been previously verified, other segments in the MHT can be verified with implicit authentication by the hash values maintained in AHT, where the hash values are from the witness field or generated while computing the root hash value of MHT. If they do not have any authenticated hash values related to $m_i^\dagger$ in their AHT, they operate MHT-based signature verification to authenticate the received $m_i^\dagger$. At this time, witness information is used to compute the root hash value of MHT. When the received segment $m_i^\dagger$ is proven to be valid, w hash values ($h_{iw} \parallel \ldots \parallel h_{(i+1)w-1}$) in its data field and hash values which have been generated for computing a root hash value of MHT are stored in the AHT for verifying segments in Data part and the other segments in Meta part.

After correctly verifying Meta part segments, requesters send Interest packets for a Data part segment. Since requesters already maintain the AHT containing authentication information for verifying Data part segments, the received segments in Data part can be verified just with hash value comparison. For example, when requesters receive a Data part segment, they compute the hash value of the segment and search the same hash value from the AHT. If there is no identical value, then the segment is proven to be faked or modified during transmission. Then the requesters send an Interest packet for retransmission.

3.3.2. Consideration of Transmission Flow Control. Like TCP flow control and the sliding window, several flow control mechanisms have been recently proposed for improving the content forwarding performance in CCN [24, 25]. For example, requesters can send several Interest packets without waiting corresponding segments or a publisher can transmit several segments when he/she receives an Interest packet. TLDA can work fairly well with these kinds of advanced transmission mechanisms. In TLDA, because of the application of MHT-based signing, the Meta part segments can be verified individually without any dependence on other segments, and out-of-order transmission or transmission loss does not affect the operation of TLDA with any of transmission mechanisms. In addition, in TLDA, the verification of Data part segments is not affected by transmission errors as long as Meta part segments are correctly received and verified, since Data part segments can be verified just with hash value lookup and comparison. On the other hand, hash chain-based segment authentication mechanism from [10] does not work properly with these advanced transmission mechanisms since it requires sequential arrival of segments, which is typically not guaranteed in CCN using advanced transmission mechanisms.

4. Correctness, Security, and Performance Analysis

In this section, we analyze various features of TLDA including its correctness, security, and performance. We will give

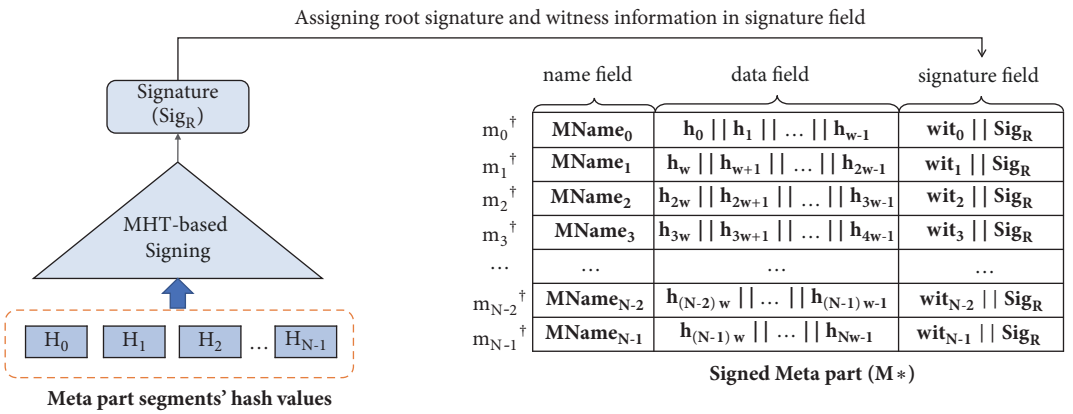

Figure 4: Meta part encoding: MHT-based signing. (For simple description, we omit signed info field in each segment in this figure).

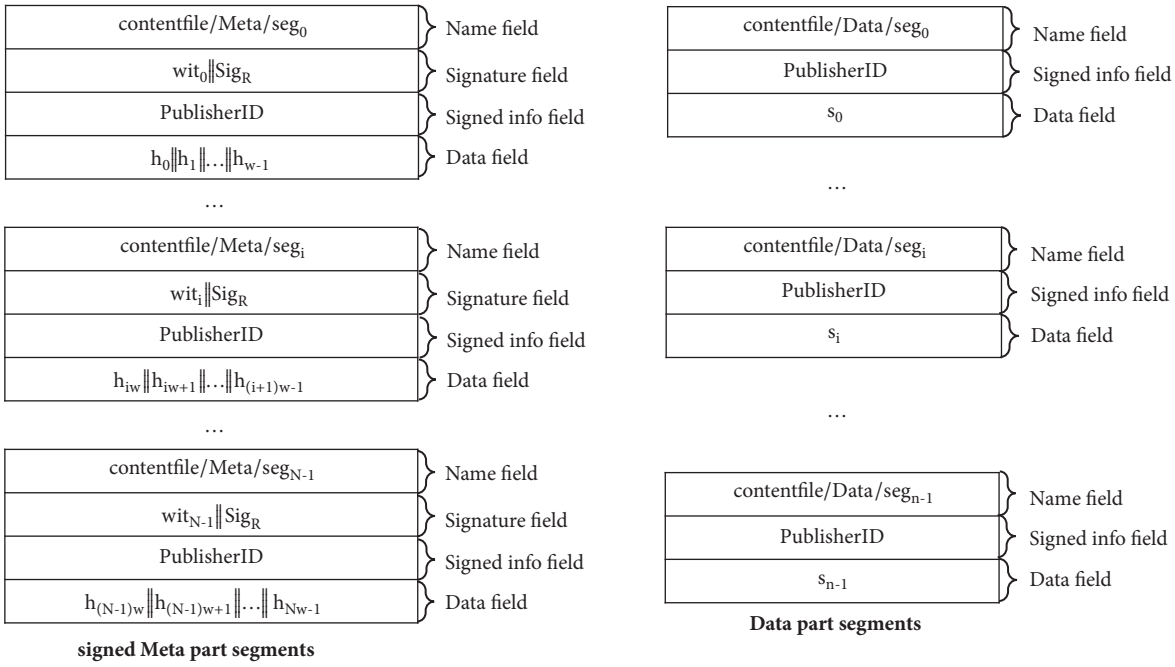

Figure 5: Example of the proposed format for segments in encoded Data part and encoded Meta part.

experimental results in the next section, and thus only theoretic aspects will be treated here. From now, we will examine the correctness of TLDA, discuss security issues of TLDA, and give theoretic performance.

4.1. Correctness. Note that TLDA is correct if a legitimately generated packet can be verified to be valid without exception. Based on the definition, we can prove the correctness of TLDA as follows.

Theorem 1. *In TLDA, any correctly generated packet can be verified without exception. Additionally, TLDA also works under transmission errors and out-of-order transmissions.*

Proof. The verification procedure in TLDA is composed of simple operations such as the verification of signature and some of hash evaluations, and the procedure is composed of two phases in sequence. For each phase, we need different conditions for the correctness. In the first phase, we deal with Meta part segments, and their correctness is identical with that of MTH-based packet verification since we use MHT-based approach for the authenticity of Meta part segments. Thus we can see that the first phase works correctly as ordinary MHT-based scheme does. Recall that we receive Data part segments after receiving all Meta part segments. It is easy to see the correctness of the verification for Data part segments since their correctness can be verified by comparing the hash values of them with Meta part segments. Therefore, we can see that the proposed authentication technique works correctly. Moreover, our technique works correctly even if there are any transmission errors or out-of-order transmissions since each Meta part segment can be

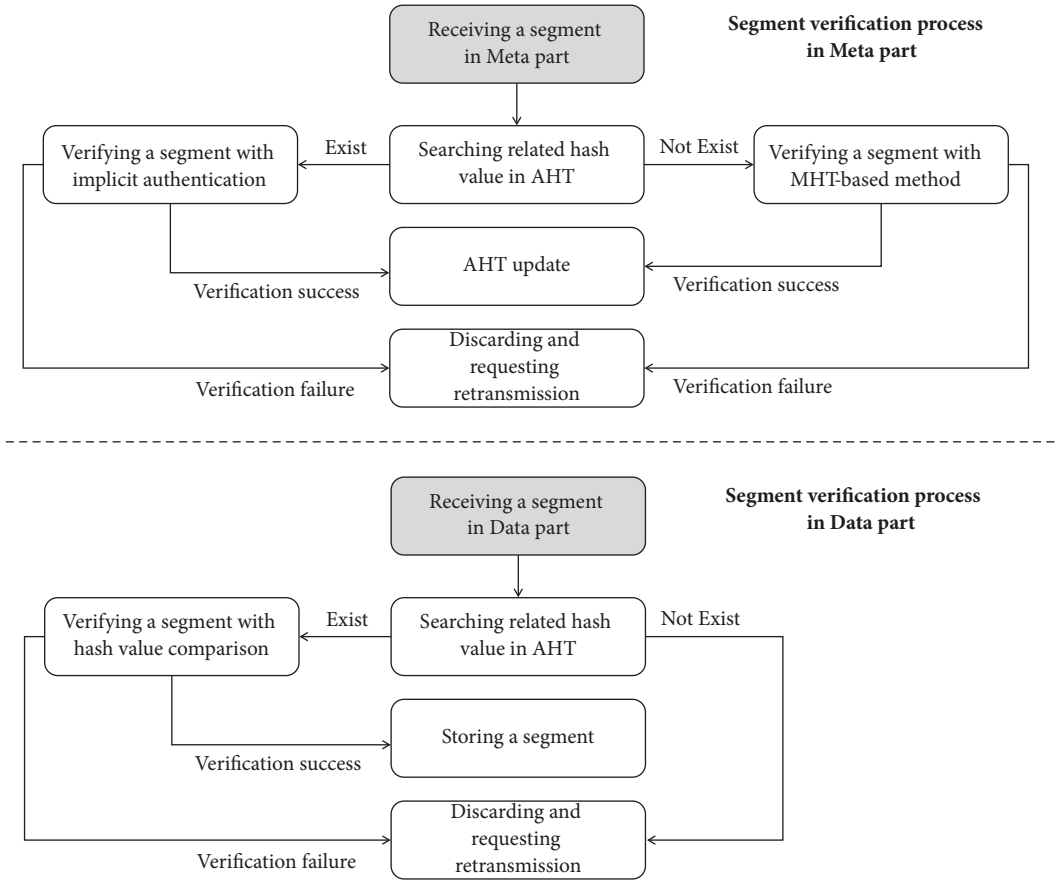

Figure 6: Proposed content segment verification process.

verified independently in the first phase because of MHT and Data part segments can be verified individually in the second phase. □

4.2. Security Analysis. Note that the goal of this work is to give an efficient method for enhancing the performance of authentication mechanism without modifying the original architecture of CCN, and thus the use of our authentication mechanism does not influence on the fundamental features of CCN. For example, we can expect the robustness against DoS/DDoS since the security feature can be guaranteed because of a basic property so-called *packet aggregation*. Moreover, because of the use of in-network cache memory in routers, we can expect higher accessibility as in the original CCN architecture.

In the same context, we will focus on the security features that have been guaranteed by the packet authentication mechanism, though there are a number of security issues (in [3], DoS/DDoS, content poisoning, cache pollution, secure naming and routing, application security, and other general security concerns have been summarized as threats against ICN techniques. An authentication mechanism could be used to protect other attacks, but the fundamental goal of an authentication mechanism in CCN is to guarantee the security against the content poisoning attack) since only the authentication mechanism is considered in our approach. Thus, before examining the security of TLDA, we want to emphasize that the goal of this work is to guarantee the security against content poisoning attack with improved performance. Since an adversary can mount content poisoning attacks only if he/she can generate a valid packet, a forgery, we will measure the security of our protocol against the forgery attacks.

The security of TLDA depends on the following two assumptions:

(1) the unforgeability of underlying signature scheme;

(2) the collision resistance of a hash function used in MHT.

In the following theorem, we will prove that the security features of underlying techniques guarantee the security of TLDA against the forgery attacks.

Theorem 2. *TLDA provides resistance against forgery as long as the underlying signature scheme is unforgeable and it is computationally hard to modify any data blocks included in an underlying MHT for Meta part segments.*

Proof. It is well-known that two assumptions mentioned above are valid since a number of signature schemes and hash functions are used in practice with the security features and

we will use suitable schemes among the existing schemes. Thus, for security proof, we can assume that

(i) underlying signature scheme is unforgeable;

(ii) any data blocks in MHT cannot be modified because of the collision resistance of underlying hash function.

As a concrete instantiation of our mechanism, we consider RSA-PSS [15] as underlying signature scheme, whose security has been proven under formal security model. We also use SHA-256 [16] for MHT, which implies that the robustness of MHT is guaranteed by the collision resistance of SHA-256. Then we can guarantee the above two propositions by the use of RSA-PSS and SHA-256 (for MHT).

Recall that the protocol is composed of two phases. In the first phase, a set of segments for Meta part will be transmitted to the requester, and the segments for Data part will be sent to requester after the first phase finished. Thus, we can classify a forged segment into two cases according to the phase.

To succeed in breaking the security of TLDA in the first phase, the adversary should generate a forged data packet for Meta part. A valid data packet is composed of a signature for the root of MHT computed from a set of hash values of Meta part and a set of authentication sibling path. Since the adversary's goal is to generate a valid data packet for a new message, at least one value should be different from the existing data packets. The different part would be the signature or some hash values. If the forgery includes a signature that was not generated before, the adversary succeeds in breaking the unforgeability of the underlying signature scheme. Otherwise, if the forgery includes a different sibling path which is valid for the same root of MHT, the adversary breaks the collision resistance of the underlying hash function. In both cases, the existence of a forgery implies that the security of one of the underlying schemes has been broken. Since we assume that the underlying signature scheme is unforgeable and the hash function (used in MHT) has collision resistance, the existence of forgery contradicts the assumptions, and thus we can say that an adversary cannot generate a valid forgery for TLDA when the underlying schemes are secure.

In the second phase, a segment for Data part can be verified to be valid only if the hash value of the data included in the segment is identical with a hash value already sent to the requester. To generate a segment, an adversary should find a data whose hash value is identical with a hash value that is already computed and transmitted to the requester. In this case, the adversary should break the collision resistance of the underlying hash function, which is not possible because of the assumption. Therefore, the adversary cannot generate a forgery in the second phase.

As we have shown in the above, by two assumptions, we can see that the proposed scheme is secure against forgery attacks. □

4.3. Performance Analysis. In a theoretical performance analysis, **S**, **V**, and **H** mean signature signing operation, signature verification operation, and hash operation, respectively.

Theorem 3. *The total cost of generating authentication information for a content of n segments in TLDA is* $\{\mathbf{S} + (n + 2N - 1)\mathbf{H}\}$.

Proof. The authentication generation process in TLDA consists of two phases: Data part encoding and Meta part encoding. In Data part encoding, n hash operations are required to generate n segments' hash values. In Meta part encoding, MHT-based signing requires $(2N-1)$ hash operations (N for leaf hash value computation and $N-1$ for MHT construction) and a signing operation. Thus, the total cost for generating authentication information in TLDA is $\{\mathbf{S} + (n + 2N - 1)\mathbf{H}\}$. □

Theorem 4. *The cost of verifying a content composed of n segments in TLDA is* $\{\mathbf{V} + (N \log_2 N + N + n)\mathbf{H}\}$.

Proof. Without implicit authentication using AHT, the content requester verifies segments with MHT-based signature verification, which requires the computation of a root hash value and signature verification per segment. Thus verifying N Meta part segments with MHT-based signature verification requires $N\{\mathbf{V} + (\log_2 N + 1)\mathbf{H}\}$ ($\mathbf{V}$ is for signature verification and $(\log_2 N + 1)\mathbf{H}$ for computing a root hash value). However, TLDA utilizes the property of MHT's implicit authentication with application of the AHT. In other words, the cost for MHT-based signature verification can be reduced by making full use of previously computed and authenticated hash values from the process for computing the root hash value of the MHT. This requires single signature verification and $(\log_2 N + 1)\mathbf{H}$ for each segment in the MHT. With implicit authentication, the cost for verifying a segment is between $\mathbf{H}$ and $(\log_2 N + 1)\mathbf{H}$, and it depends on the position of the segment in the MHT. Thus, we conservatively compute the number of hash computations with $(\log_2 N+1)\mathbf{H}$ for verifying a Meta part segment with implicit authentication since TLDA assumes out-of-order transmission. Since $n\mathbf{H}$ is required for verifying n Data part segments, the total cost is $\{\mathbf{V} + (N \log_2 N + N + n)\mathbf{H}\}$. □

Table 2 compares theoretical computation and communication overhead of TLDA with original CCNx at sender and receiver sides. Computation overhead is the additional computation incurred from the application of TLDA when transmitting a content of n segments at sender and receiver side, respectively. Similarly, communication overhead means the additional information required to authenticate the content composed of n segments. We assume that a content consists of $n(= w \cdot N)$ segments and MHT in N size is applied to CCNx and TLDA. Thus, CCNx makes w MHT groups for n segments and applies MHT-based signing for w groups of N segments. Since CCNx applies MHT-based signing to all segments of the content, it requires $\{w(\mathbf{S} + (2N - 1)\mathbf{H})\}$ and $\{w(\mathbf{V} + N(\log_2 N + 1)\mathbf{H})\}$ for signing n segments and verifying them in total. Unlike CCNx, which signs all data segments, since TLDA applies MHT-based signing only to Meta part of N segments, it requires single signature signing at sender sides and single verification at receiver sides, respectively. If we assume that the sizes of

Table 2: Theoretical computational overhead and communication overhead comparison. (We assume that a content consists of $n(= w \cdot N)$ segments and we apply MHT of N size for both mechanisms. $|\cdot|$ symbol denotes the output size of corresponding operation result.)

Method	Computational overhead		Communication overhead
CCNx	Sender	$\{w(\mathbf{S} + (2N-1)\mathbf{H})\}$	$w \cdot N(\lvert\mathbf{S}\rvert + \log_2 N\lvert\mathbf{H}\rvert)$
	Receiver	$\{w(\mathbf{V} + N(\log_2 N + 1)\mathbf{H})\}$	
TLDA	Sender	$\{\mathbf{S} + (n + 2N - 1)\mathbf{H}\}$	$N(\lvert\mathbf{S}\rvert + \log_2 N\lvert\mathbf{H}\rvert)$
	Receiver	$\{\mathbf{V} + (N \log_2 N + N + n)\mathbf{H}\}$	

content are 128Mbytes, its segment is 4Kbytes, and hash value is 32bytes, and n, N, and w are 32,768, 256, and 128, respectively. On this assumption, CCNx requires $\{128\mathbf{S} + 65,408\mathbf{H}\}$ and $\{128\mathbf{V} + 294,912\mathbf{H}\}$ at sender and receiver sides, respectively, while with TLDA a sender and a receiver consume $\{\mathbf{S}+33,279\mathbf{H}\}$ and $\{\mathbf{V}+35,072\mathbf{H}\}$, respectively. With respect to communication overhead, TLDA requires only $N(|\mathbf{S}| + \log_2 N|\mathbf{H}|)$ in total, which is almost $(1/w)$ reduced overhead compared with the original CCNx consuming $w \cdot N(|\mathbf{S}| + \log_2 N|\mathbf{H}|)$. Since both TLDA and the original CCNx utilize MHT-based signing, they attach a root signature and authentication sibling path in each segment, which requires $(|\mathbf{S}|+\log_2 N|\mathbf{H}|)$. However, in case of TLDA, only N segments in Meta part are signed while the original CCNx signs all of $n(= w \cdot N)$ segments.

5. Experimental Results

This section compares TLDA with the original CCNx equipped with MHT-based segment authentication in terms of computation and communication overhead. We have implemented the prototype of TLDA in CCNx [8] which is an open source project implementing basic CCN protocols and several applications. We have implemented the prototype in Java and integrated it into CCNx Java source.

5.1. Testbed Setup. To demonstrate actual performance of TLDA, we have composed a testbed, where a content requester receives a content from a content publisher by utilizing ccnputfile and ccngetfile test applications, which are provided by basic applications in CCNx on the 1Gbps Ethernet LAN environment. As a content publisher and a content requester, we utilize Laptops with Intel i5-2467M 1.6GHz CPU and 4GB RAM.

In general, content delivery in CCN is requester-driven. In other words, if a content requester broadcasts Interest packets containing the name of the content that he/she wants to receive, network nodes (including a content publisher) send data packets corresponding to the received Interest packets. A segment number is appended to a content name to identify Interest packets and Data packets. In our testbed, when the content publisher receives an Interest packet, it generates a corresponding Data packet, which is composed of content name field, signature field, signed info field, and data field. At this time, the content publisher generates segment authentication information and stores it at the signature field of the Data packet. Next, the content publisher sends the generated Data packet. When a content requester receives the Data packet, he/she verifies it. If the Data packet is proven to be valid, the content requester sends Interest packet for the next segments. Otherwise, the content requester requests retransmission for the valid segment. This process continues until the content requester finishes receiving all segments in the content.

As to generating segment authentication information, TLDA works differently from the MHT-based segment authentication mechanism in CCNx. In case of CCNx, it applies MHT-based signing mechanism to all segments in the content. That is, it reads k (the size of leaf nodes in MHT) segments from the content file at a time and constructs an MHT for the segments and then finally generates root signature. CCNx repeats this MHT-based signing process by units of k segments until it finishes the process for generating segment authentication information while responding to Interest packets from the content requester, simultaneously. On the other hand, TLDA generates hash values for Data part segments and constructs Meta part then applies MHT-based signing only to the segments in Meta part sequentially. As aforementioned in Section 3.2, even though hash value computations in Data part encoding and Meta part encoding can be processed in offline because of the deterministic property of hash function, we include the time for all hash value computations for the measurement. In our testbed, the segments in Meta part are firstly transmitted from the content publisher to the content requester and after completing the transmission of Meta part, segments in Data part are delivered. Meta part segments are authenticated by MHT-based signature verification with implicit authentication and Data part segments are authenticated just with hash value comparison. Since CCNx does not support implicit authentication using AHT mechanism described in Section 2.2, where segments can be verified by using already authenticated hash values, in its mechanism, we have implemented the functionality of AHT to CCNx for fair comparisons.

TLDA utilizes RSA-PSS and SHA-256 as its underlying signature algorithm and hash function. Regarding RSA key size, it supports not only 1024-bit for legacy systems but also 2048-bit for satisfying the currently recommended security strength. We set the size of content as 128Mbytes. Even though original CCNx uses 4Kbyte segment size, we have applied various segment sizes such as 1Kbytes, 2Kbytes, and 4Kbytes considering network environments where the maximum packet size is lower than 4Kbytes.

5.2. Computational Performance and Overhead Analysis. To evaluate the overhead incurred from segment authentication mechanisms, we have first set NULL mode, which is a version of CCNx without data segment authentication, as a baseline

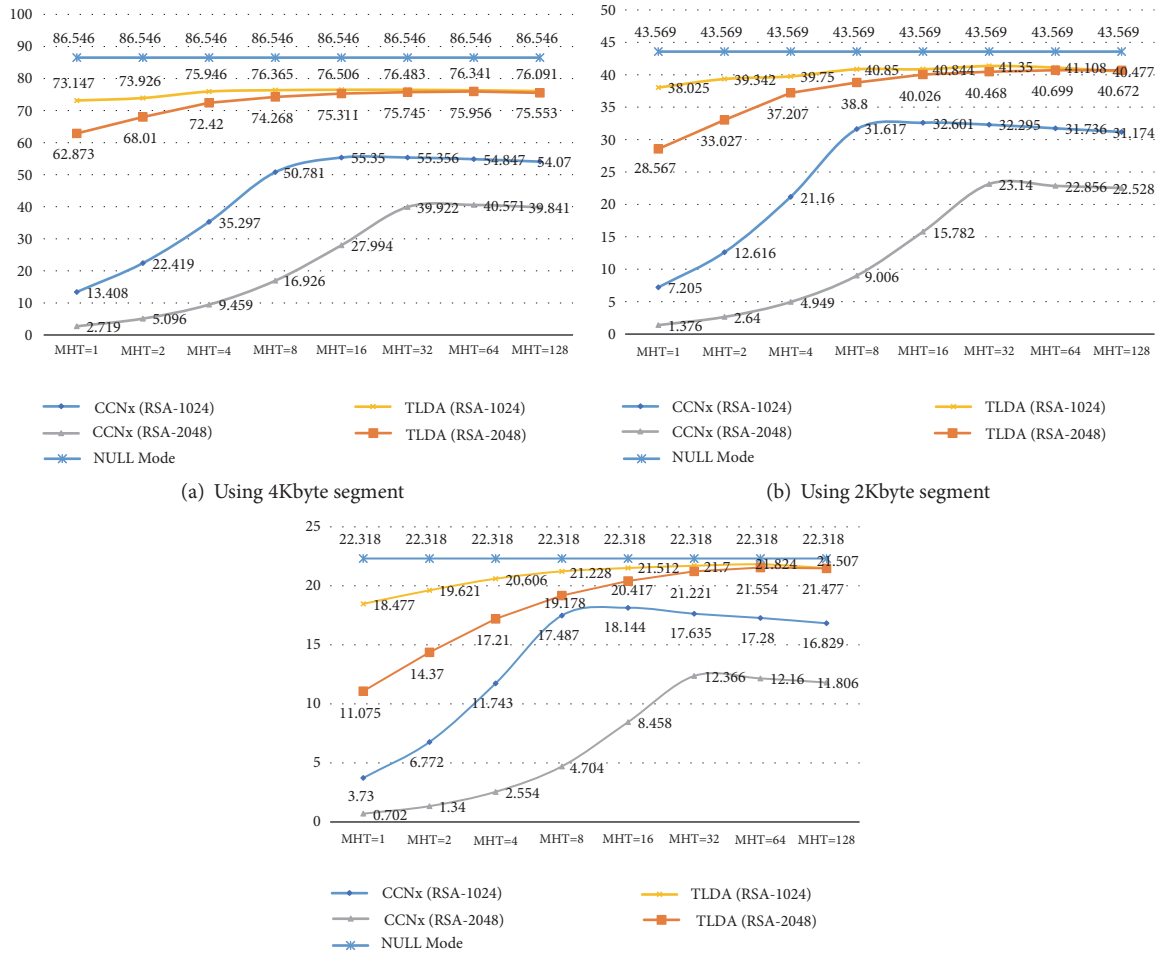

FIGURE 7: Transmission performance comparison when transmitting a 128Mbyte content with segment size of 4Kbytes, 2Kbytes, and 1Kbytes. (x-axis is MHT size applied to both CCNx and TLDA and y-axis is Mbps.)

for comparison with TLDA and CCNx, and measured the performance of content transmission (it means transmission throughput). Then we have taken a measurement of that of TLDA and CCNx with various MHT sizes from 1 to 128. TLDA applies MHT-based signing mechanism to Meta part segments while CCNx applies MHT-based signing to all segments of the content.

Figure 7(a) shows the transmission throughput of TLDA, CCNx, and NULL mode when the size of segment is 4Kbytes. NULL mode without segment authentication provides 86.546 Mbps. Even though NULL mode does not involve the process of segment authentication, it has a limited transmission performance. That is because in CCNx protocol execution content name encoding/decoding procedure is significantly time consuming [26, 27]. TLDA with 1024-bit (resp., 2048-bit) key size provides the best performance of 76.506 Mbps (resp., 75.956 Mbps) when MHT size is 16 (resp., 64) while CCNx with 1024-bit (resp., 2048-bit) key size offers the best performance of 55.356 Mbps (resp., 40.571 Mbps) when MHT size is 32 (resp. 64). In TLDA, the performance gap between using RSA-1024 and RSA-2048 is negligible compared with CCNx since MHT-based signing is applied only to Meta part segments in TLDA while CCNx applies MHT-based signing to all segments in the content. From Figure 8, we have found that TLDA provides around 1.382 (resp., 1.872) times better transmission throughput when using RSA-1024 (resp., RSA-2048) signing compared with CCNx. Figure 8(a) compares delay overhead incurred from TLDA and CCNx when transmitting a 128Mbyte content in units of 4Kbyte segment. Depending on the applied MHT size, TLDA induces slight overhead of 11.601%–15.482% (resp., 12.236%–27.353%) when RSA-1024 (resp., RSA-2048) is used, while CCNx incurs severe overhead of 36.039%–84.508% (resp., 53.122%–96.858%) when using RSA-1024 (resp., RSA-2048). We compute the delay overhead from TLDA and CCNx as [{(*throughput of NULL*) − (*throughput of TLDA* or *CCNx*)}/(*throughput of NULL*)]%.

Figures 7(b) and 8(b) compare the transmission throughput and transmission delay incurred by TLDA and CCNx when transmitting a 128Mbyte content in units of 2Kbyte segment, respectively. It is noticeable that the transmission performance of NULL mode decreases about in half. This is

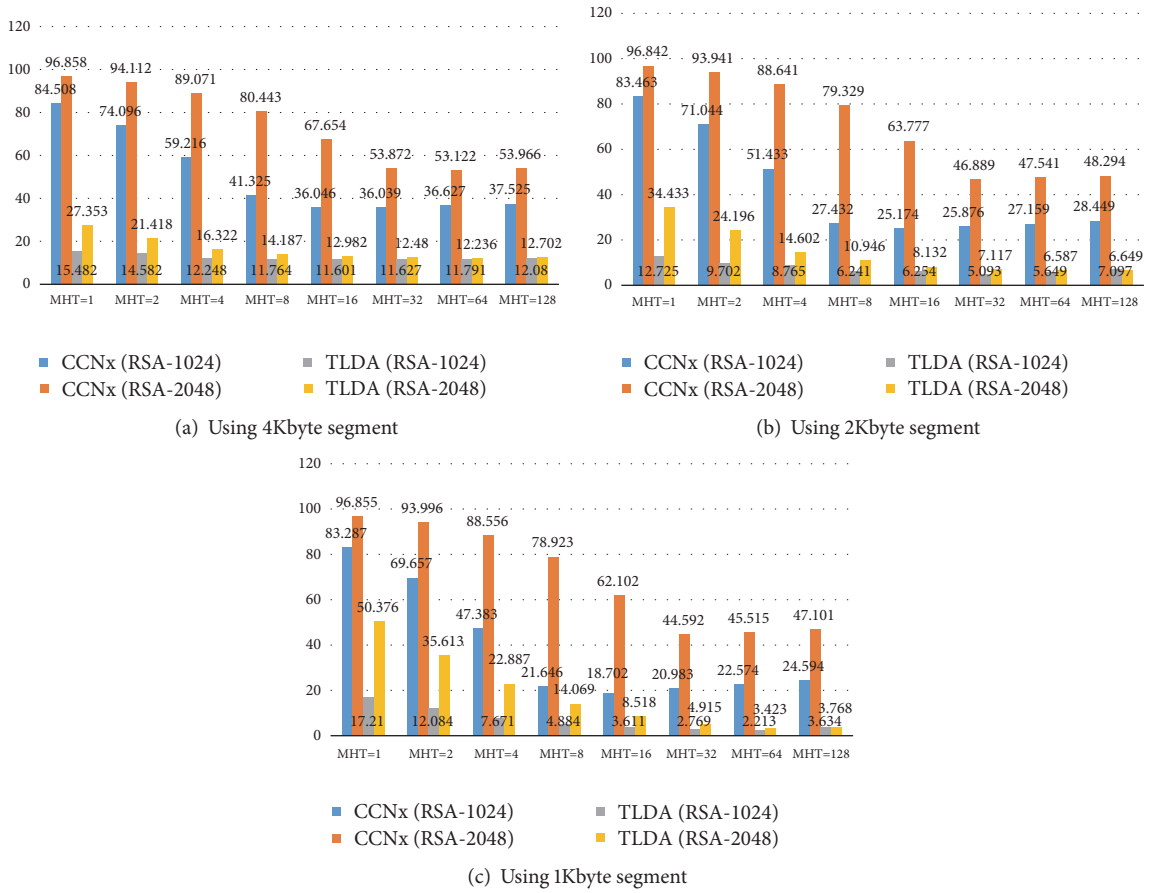

FIGURE 8: Transmission delay overhead comparison when transmitting a 128Mbyte content with segment size of 4Kbytes, 2Kbytes, and 1Kbytes. (x-axis is MHT size applied to both CCNx and TLDA and y-axis is introduced transmission delay overhead (%).)

because the number of Interest packets and Data packets is doubled. Even though the size of data field is reduced from 4Kbytes to 2Kbytes, the cost for generating a Data packet is equal regardless of the size of data field. The same principle holds in the cost for generating Interest packets. Thus, the number of Interest packets and Data packets is the critical factor affecting the overall transmission performance under the circumstance where content name encoding/decoding mainly affects the performance of processing Interest packets and Data packets. TLDA with 1024-bit (resp., 2048-bit) key size provides the best performance of 41.35 Mbps (resp., 40.699 Mbps) when MHT size is 32 (resp., 64) while CCNx with 1024-bit (resp., 2048-bit) key size offers the best performance of 32.601 Mbps (resp., 23.14 Mbps) when MHT size is 16 (resp., 32). TLDA provides almost 1.268 (resp., 1.758) times better performance when using RSA-1024 (resp., RSA-2048) signing compared with CCNx. From the Figure 8(b), we have found out that, depending on the applied MHT size, TLDA brings small delay overhead of 5.093%–12.725% (resp., 6.587%–34.433%) when RSA-1024 (resp., RSA-2048) is used while CCNx incurs heavy delay overhead of 25.174%–83.463% (resp., 46.889%–96.842%) when using RSA-1024 (resp., RSA-2048).

It is worthy of notice that the delay overhead from segment authentication when the size of segment is 2Kbytes decreases compared with that when the size of segment is 4Kbytes. This is because even though the ratio of the number of cryptographic operations per segment is identical without regard to the segment size, the time consumed for processing a cryptographic operation on 2Kbyte segment is lower than that on 4Kbyte segment. Table 3 shows the time of cryptographic operations depending on the segment sizes. The less the segment size, the less the time consumed. This results from the property of SHA-256 hash algorithm. It is known that hash algorithm gets variable length of input and generates fixed length of output. SHA-256 also gets any length of data by units of 512-bit and generates 256-bit message digest. For example, if the size of data is 2048-bit, SHA-256 inputs the data and divides it into four blocks and sequentially executes internal hash calculations. Thus if the size of input data increases, the time for calculating its hash value increases at the rate of the data input increase.

Figures 7(c) and 8(c) compare the transmission throughput and delay overhead of TLDA and CCNx when transmitting a 128Mbyte content by units of 1Kbyte segment. TLDA with 1024-bit (resp., 2048-bit) key size provides the best performance of 21.824 Mbps (resp., 21.554 Mbps) while CCNx with 1024-bit (resp., 2048-bit) key size offers the best performance of 18.144 Mbps (resp., 12.366 Mbps). TLDA offers around 1.203 (resp. 1.743) times better performance

Table 3: Time comparison for RSA signing, verifying, and hashing to a segment of 4Kbytes, 2Kbytes, and 1Kbytes. (Timings are the average of one million executions.) RSA signature operation makes use of SHA-256 for message hashing.

	Segment Size		
	4Kbytes	**2Kbytes**	**1Kbytes**
RSA 1024 Signing	1.9236 *ms*	1.8659 *ms*	1.7832 *ms*
RSA 1024 Verifying	0.3508 *ms*	0.2955 *ms*	0.2624 *ms*
RSA 2048 Signing	11.2315 *ms*	10.9907 *ms*	10.8255 *ms*
RSA 2048 Verifying	0.7487 *ms*	0.6983 *ms*	0.6753 *ms*
SHA-256	0.05071 *ms*	0.02556 *ms*	0.01329 *ms*

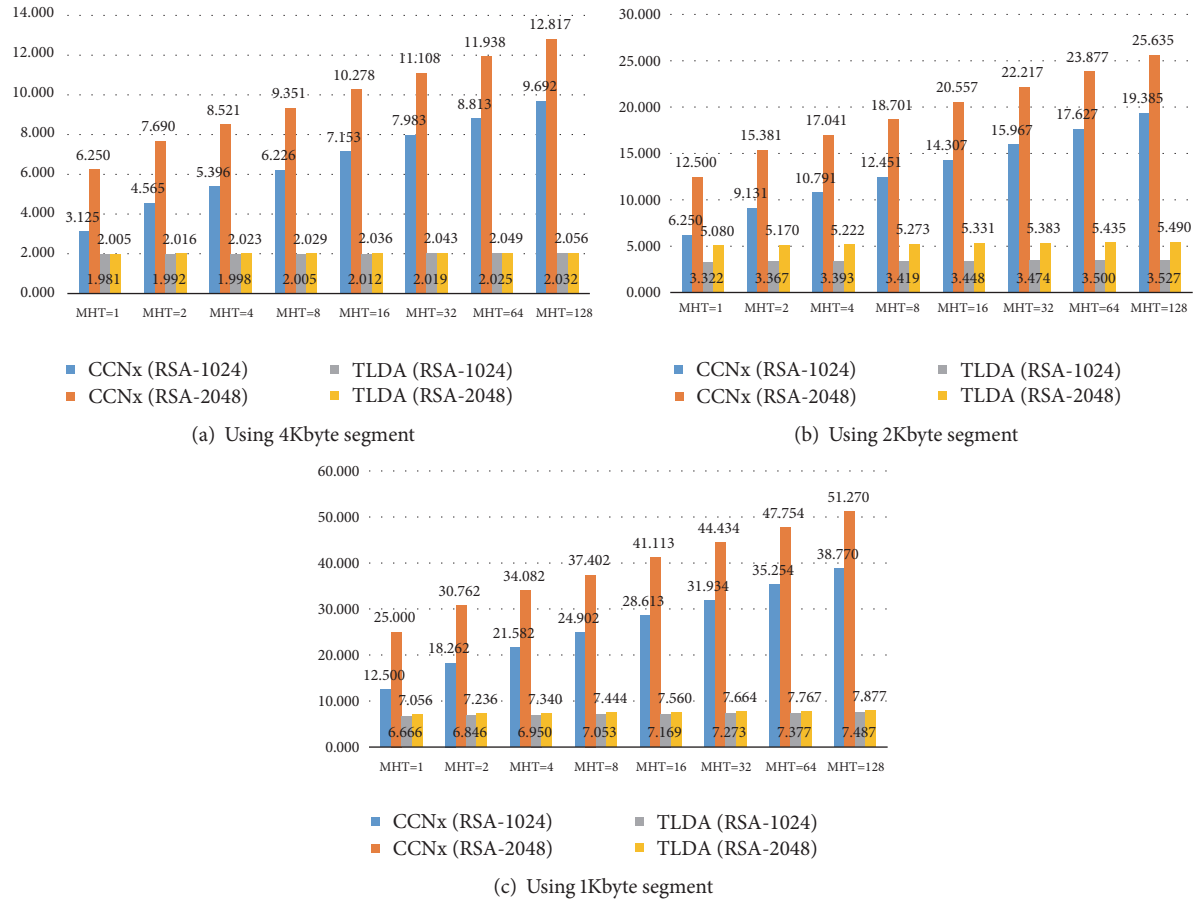

Figure 9: Communication overhead comparison when transmitting a 128Mbyte content with segment size of 4Kbytes, 2Kbytes, and 1Kbytes. (x-axis is MHT size applied to both CCNx and TLDA and y-axis is the introduced communication overhead.)

when using RSA-1024 (resp., RSA-2048) signing compared with CCNx. From Figure 8(c), we have found that, depending on the applied MHT size, TLDA introduces little overhead of 2.213%–17.21% (resp., 3.423%–50.376%) when RSA-1024 (resp., RSA-2048) is used while CCNx generates heavy overhead of 18.702%–83.287% (resp., 44.592%–96.855%) when using RSA-1024 (resp., RSA-2048).

5.3. Communication Overhead Analysis. In this section, we analyze the communication overhead raised from TLDA and CCNx. We have referred to the ratio of the size of additional authentication information to data field size as communication overhead. The reason for excluding content name and signed info fields in the measurement is that the sizes of the fields are variable while the size of data field is constant. Figure 9(a) compares TLDA and CCNx regarding communication overhead when transmitting a 128Mbyte content and the segment size is 4Kbytes. Since CCNx applies MHT-based signing to all segments in the content, the amount of additional authentication information increases linearly depending on the applied MHT size. This is because the size of witness including hash values to compute the root hash value increases linearly, even though the signature size is identical regardless of MHT size. In case of TLDA,

it is not heavily affected by the applied MHT size since MHT-based signing is applied only to segments in Meta part. In the previous section, when transmitting a content in units of 4Kbyte segment, MHT size 32 (resp., 64) shows the best performance for RSA-1024 (resp., RSA-2048) signing in CCNx. CCNx equipped with MHT size 32 (resp., 64) incurs 7.983% (resp., 11.938%) of communication overhead for authenticating segments with RSA-1024 (resp., RSA-2048). However, TLDA induces significantly reduced overhead of around 2% compared with CCNx. Figure 9(b) compares TLDA and CCNx when transmitting a 128Mbyte content with units of 2Kbyte segment. Even though CCNx offers the best performance when MHT size 16 (resp., 32) for RSA-1024 (resp., RSA-2048), it requires 14.307% (resp., 22.217%) of communication overhead. However, TLDA requires only 3.474% (resp., 3.5%) of communication overhead when it provides the best performance with MHT size 32 (resp., 64) for RSA-1024 (resp., RSA-2048). Figure 9(c) shows communication overhead of TLDA and CCNx when transmitting a 128Mbyte content in units of 1Kbyte segment. When MHT size is 16 (resp., 32) for RSA-1024 (resp., RSA-2048), CCNx provides the best performance, but it causes 28.613% (resp., 44.434%) communication overhead. TLDA brings only about 7.487% (resp., 7.877%) of communication overhead when providing the best performance with MHT size 64 for RSA-1024 and RSA-2048.

6. Conclusion and Future Work

In this paper, we have proposed TLDA, which is an efficient Two-Layered Data Authentication mechanism for CCN. For efficiency of computation and communication, TLDA newly introduces the concept of authentication Meta part consisting of data segments' hash values. To a considerable extent, TLDA not only reduces the computation and communication overhead compared with CCN's basic authentication method, but also provides robustness against transmission loss and out-of-order transmission. We have presented the details of TLDA covering the processes of generating segment authentication information and verifying segments. In order to demonstrate the effectiveness of TLDA, we have prototyped TLDA in CCNx library. Experimental results show that TLDA provides much enhanced performance in computation and communication compared with existing segment authentication mechanisms in CCNx. TLDA can be efficiently used for authenticating constant contents such as VOD, installation files, and large documents files.

In future works, we have a plan to apply Forward Error Correction (FEC) techniques [28–30], e.g., erasure and network codes, for efficient and reliable delivery of Meta part segments in broadcast and multicast environments. With the application of FEC to TLDA, requesters can recover the original Meta part even when some parts of the Meta part are missed or not arrived. We expect that FEC-based TLDA can provide more enhanced performance of efficient and reliable delivery of Meta part segments. Furthermore, we will extend TLDA such that it provides protection against content pollution attack [31–34] on the network nodes. In the future works, we will verify the advantage of our mechanism with large scale scenario by using ndnSIM [35] which is a recently developed NDN simulator.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

This work was supported by Electronics and Telecommunications Research Institute (ETRI) grant funded by the Korean Government [18ZH1200, Core Technology Research on Trust Data Connectome].

References

[1] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, "Networking named content," in *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies (CoNEXT '09)*, pp. 1–12, December 2009.

[2] T. Koponen, M. Chawla, B.-G. Chun et al., "A data-oriented (and beyond) network architecture," in *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, And Protocols for Computer Communications (SIGCOMM '07)*, pp. 181–192, 2007.

[3] R. Tourani, S. Misra, T. Mick, and G. Panwar, "Security, privacy, and access control in information-centric networking: a survey," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 1, pp. 566–600, 2017.

[4] "NetInf project," http://www.netinf.org.

[5] "PSIRP project," http://www.psirp.org.

[6] Diana Smetters and Van Jacobson, "Securing network content," PARC Technical Report, 2009.

[7] "The content centric networking (CCNx) project," https://named-data.net.

[8] "CCNx project," http://github.com/ProjectCCNx/ccnx.

[9] Jeff Burke, Alex Horn, and Alessandro Marianantoni, "Authenticated lighting control using named data networking," NDN Technical Report NDN-011, 2012.

[10] J. Burke, P. Gasti, N. Nathan, and G. Tsudik, "Securing instrumented environments over content-centric networking: the case of lighting control and NDN," in *Proceedings of the IEEE INFOCOM 2013 Workshop on Emerging Design Choices in Named-Oriented Networking*, pp. 394–398, IEEE, Turin, Italy, April 2013.

[11] M. Baugher, B. Davie, A. Narayanan, and D. Oran, "Self-verifying names for read-only named data," in *Proceedings of the IEEE INFOCOM 2012 Workshop on Emerging Design Choices in Named-Oriented Networking*, pp. 274–279, USA, March 2012.

[12] Ilya Moiseenko, "Fetching content in named data networking with embedded manifests," NDN Technical Report NDN-0025, 2014.

[13] T. Refaei, M. Horvath, M. Schumaker, and C. Hager, "Data authentication for NDN using hash chains," in *Proceedings of the 20th IEEE Symposium on Computers and Communication, ISCC 2016*, pp. 982–987, 2016.

[14] Q. Li, X. Zhang, Q. Zheng, R. Sandhu, and X. Fu, "LIVE: Lightweight integrity verification and content access control

for named data networking," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 2, pp. 308–320, 2015.

[15] RSA-PSS (Probabilistic Signture Scheme), "ISO/IEC 14888-2:2008 Information technology-Security techniuqes-Digital signatures with appendix-Part 2: Integer factorization based mechanisms," 2008.

[16] SHA-256, *FIPS 180-4: Secure Hash Standard (SHS)*, 2015.

[17] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang, "DoS and DDoS in named data networking," in *Proceedings of the 22nd International Conference on Computer Communications and Networks, ICCCN 2013*, pp. 1–7, IEEE, Nassau, Bahamas, August 2013.

[18] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Proceedings of the Conference on the Theory and Application of Cryptographic Techniques on Advances in cryptology (CRYPTO'87)*, vol. 293 of *Lecture Notes in Computer Science*, pp. 369–378, Springer, Berlin, Germany, 1987.

[19] R. C. Merkle, "A certified digital signature," in *Proceedings of the Conference on the Theory and Application of Cryptology on Advances in cryptology (CRYPTO'89)*, vol. LNCS 435 of *Lecture Notes in Computer Science*, pp. 218–238, Springer, New York, NY, USA.

[20] C. K. Wong and S. S. Lam, "Digital signatures for flows and multicasts," *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 502–513, 1999.

[21] Y.-C. Hu, M. Jakobsson, and A. Perrig, "Efficient constructions for one-way hash chains," in *Proceedings of the Third International Conference on Applied Cryptography and Network Security, ACNS 2005*, vol. LNCS 3531, pp. 423–441, USA, June 2005.

[22] J. Deng, R. Han, and S. Mishra, "Secure code distribution in dynamically programmable wireless sensor networks," in *Proceedings of the 5th International Conference on Information Processing in Sensor Networks (IPSN '06)*, pp. 292–300, ACM, Nashville, Tenn, USA, April 2006.

[23] S. Hyun, P. Ning, A. Liu, and W. Du, "Seluge: Secure and DoS-resistant code dissemination in wireless sensor networks," in *Proceedings of the 2008 International Conference on Information Processing in Sensor Networks, IPSN 2008*, pp. 445–456, April 2008.

[24] D. Byun, B.-J. Lee, and Y. Park, "Adaptive interest adjustment in CCN flow control," in *Proceedings of the 2014 IEEE Global Communications Conference, GLOBECOM 2014*, pp. 1873–1877, December 2014.

[25] A. Gupta and K. Bansal, "Flow control enhancements in content centric networking," in *Proceedings of the 2015 7th International Conference on Communication Systems and Networks, COMSNETS 2015*, p. 18, January 2015.

[26] H. Yuan and P. Crowley, "Performance measurement of name-centric content distribution methods," in *Proceedings of the 2011 7th ACM/IEEE Symposium on Architectures for Networking and Communications Systems, ANCS 2011*, pp. 223-224, October 2011.

[27] H. Yuan and P. Crowley, "Experimental evaluation of content distribution with NDN and HTTP," in *Proceedings of the IEEE INFOCOM 2013 Mini-Conference*, pp. 240–244, 2013.

[28] S. Kim, R. Fonseca, and D. Culler, "Reliable transfer on wireless sensor networks," in *Proceedings of the 2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004.*, pp. 449–459, Santa Clara, CA, USA.

[29] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh, "A Survey on network codes for distributed storage," *Proceedings of the IEEE*, vol. 99, no. 3, pp. 476–489, 2011.

[30] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," *ACM SIGCOMM Computer Communication Review*, vol. 27, no. 2, pp. 24–36, 1997.

[31] M. Conti, P. Gasti, and M. Teoli, "A lightweight mechanism for detection of cache pollution attacks in Named Data Networking," *Computer Networks*, vol. 57, no. 16, pp. 3178–3191, 2013.

[32] E. G. Abdallah, H. S. Hassanein, and M. Zulkernine, "A survey of security attacks in information-centric networking," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 3, pp. 1441–1454, 2015.

[33] A. Karami and M. Guerrero-Zapata, "An ANFIS-based cache replacement method for mitigating cache pollution attacks in named data networking," *Computer Networks*, vol. 80, no. 7, pp. 51–65, 2015.

[34] H. Guo, X. Wang, K. Chang, and Y. Tian, "Exploiting path diversity for thwarting pollution attacks in named data networking," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 9, pp. 2077–2090, 2016.

[35] S. Mastorakis, A. Afanasyev, and L. Zhang, "On the evolution of ndn SIM: An open-source simulator for NDN experimentation," *ACM SIGCOMM Computer Communication Review*, vol. 47, no. 3, pp. 19–33, 2017.