sustainability

MDPI

Article

Faster Data Forwarding in Content-Centric Network via Overlaid Packet Authentication Architecture

Taek-Young Youn [1], Joongheon Kim [2], David Mohaisen [3] and Seog Chung Seo [4,*]

[1] Department of Industrial Security, Dankook University, Gyeonggi-do 16889, Korea; taekyoung@dankook.ac.kr
[2] School of Electrical Engineering, Korea University, Seoul 02841, Korea; joongheon@korea.ac.kr
[3] Department of Computer Science, University of Central Florida, Orlando, FL 32816, USA; mohaisen@ucf.edu
[4] Department of Financial Information Security, Kookmin University, Seoul 02707, Korea
* Correspondence: scseo@kookmin.ac.kr; Tel.:+82-02-910-4742

Received: 18 September 2020; Accepted: 11 October 2020; Published: 21 October 2020

Abstract: Content-Centric Networking (CCN) is one of the emerging paradigms for the future Internet, which shifts the communication paradigm from host-centric to data-centric. In CCN, contents are delivered by their unique names, and a public-key-based signature is built into data packets to verify the authenticity and integrity of the contents. To date, research has tried to accelerate the validation of the given data packets, but existing techniques were designed to improve the performance of content verification from the requester's viewpoint. However, we need to efficiently verify the validity of data packets in each forwarding engine, since the transmission of invalid packets influences not only security but also performance, which can lead to a DDoS (Distributed Denial of Service) attack on CCN. For example, an adversary can inject a number of meaningless packets into CCN to consume the forwarding engines' cache and network bandwidth. In this paper, a novel authentication architecture is introduced, which can support faster forwarding by accelerating the performance of data validation in forwarding engines. Since all forwarding engines verify data packets, our authentication architecture can eliminate invalid packets before they are injected into other CCN nodes. The architecture utilizes public-key based authentication algorithms to support public verifiability and non-repudiation, but a novel technique is proposed in this paper to reduce the overhead from using PKI for verifying public keys used by forwarding engines and end-users in the architecture. The main merit of this work is in improving the performance of data-forwarding in CCN regardless of the underlying public-key validation mechanism, such as PKI, by reducing the number of accesses to the mechanism. Differently from existing approaches that forgive some useful features of the Naive CCN for higher performance, the proposed technique is the only architecture which can support all useful features given by the Naive CCN.

Keywords: content-centric networking; data authentication; fast data forwarding; authentication infrastructure

1. Introduction

Content-Centric Networking (CCN) has recently been proposed as a means of Information-Centric Networking (ICN) to solve the existing IP-based Internet's problems such as heavy traffic increases and security weakness. Differently from the other ICN approaches, such as DONA and NetInf, CCN changes the existing networking paradigm from host-centric to data-centric.

In CCN, contents are routed by their own content name rather than their IP-address and contents are stored by the units of data segment in network nodes' storage (so-called in-network caching) during

transmission to rapidly handle future requests. CCN's name-based routing and in-network caching enable multi-path content forwarding and optimization of bandwidth usage, which resolves the current Internet's data traffic problems and DDoS security attacks [1]. Since CCN allows contents to be stored in network nodes by units of segment, it employs a public key signature as the built-in data and source authentication mechanism.

Though the use of signature guarantees built-in security to CCN data segments, it requires high computational overhead. To date, research has tried to reduce the cost of the public-key-signature-based approach for segment authentication [1–12]. PARC used an MHT-based [5,6] mechanism in CCNx. Note that CCNx is the reference implementation of CCN [1–4]. The use of MHT reduces the amount of signing and verification, but it still requires a heavy overhead compared with the scenario where segment authentication is not considered. To handle the efficiency issue, HMAC-based techniques have been introduced [9,10]. However, they are not setup-free and and the techniques need an additional method for the secure distribution of HMAC keys. Name-based techniques [7,8] have been proposed with efficient data authentication. However, they depend on specific naming roles and the reliability of names can be verified by an additional trusted party [7] or periodic broadcasting [8]. A hash chain-based mechanism was proposed for efficient data segment authentication in CCN [12]. The technique is efficient in the viewpoint of both computation and communication, but it is vulnerable to transmission error due to the basic feature of hash chain technique. In 2015, a multi-token-based method, LIVE, was proposed for efficient data integrity checking and access control in CCN nodes [11]. In the technique, the publisher uses tokens to control cached content in forwarding engines. The publisher distributes tokens to valid users. Though the technique is efficient to support sufficient performance, a content publisher should have a complete list of authorized CCN nodes and their public keys.

1.1. Contribution and Organization of the Paper

In this work, we investigate the security issues caused by the lack of packet verification in forwarding engines and review existing works under this perspective. To date, several data packet authentication mechanisms have been presented [1–3,9–11,13,14], but they all failed to achieve all of the aforementioned requirements if all forwarding engines participate in data packet verification (a detailed analysis and comparison will be described in Section 6). To support efficient packet verification in all forwarding engines, a new authentication architecture named Overlaid Packet Authentication (OPA) is designed in this work. Our authentication architecture can support all features given by the original CCN architecture. Recall that a number of improved architectures are aimed at the efficiency of packet verification in requesters, and thus they are unable to catch all features when forwarding engines also participate in the verification of data packets. In the literature, our work is the first authentication mechanism that can improve the performance of data packet verification in CCN while permitting all nodes to partipate in the verification of data packets.

This paper is organized as follows. In Section 2, we recall the basic properties of CCN and some security issues that should be considered. In Section 3, a new architecture for efficient packet verification in CCN is proposed and its security is proven in Section 4. Note that the proposed architecture is a generic construction in the sense that any signature scheme can be used. In Section 5, a concrete description is given. The performance of the proposed construction will be analyzed in Section 6. Section 7 is the conclusion.

2. Preliminaries

In this section, the basic properties of CCN are briefly recalled, with discussions regarding security-related issues.

2.1. Basic Structure of CCN

2.1.1. Packet Structures

As seen in Figure 1, there are two packet structures: the interest packet *I-Packet* and data packet *D-Packet*.

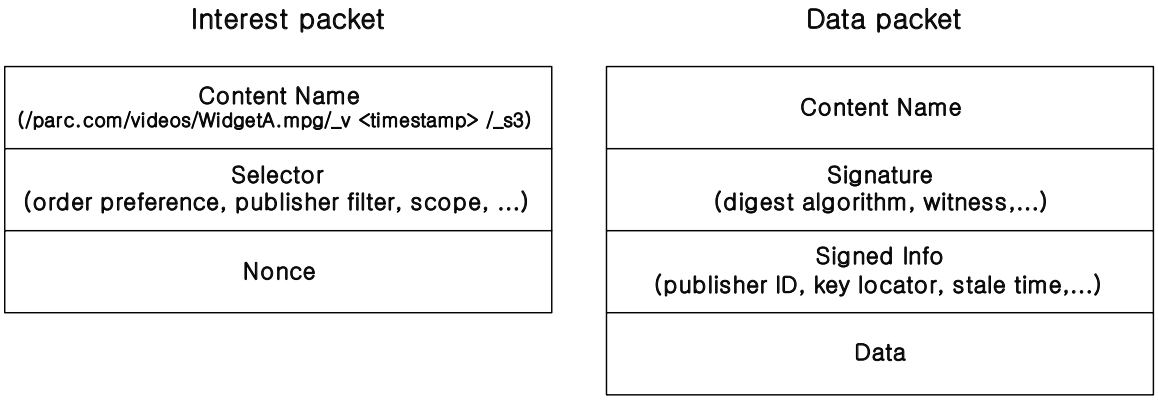

Figure 1. CCN packet types [1].

Interest Packet

An interest packet is a content request packet and it contains *content name*, *selector*, and *nonce* fields. *content name* is the name of the requested content, and typically hierarchical naming is used in CCN. *selector* indicates the condition of the required data. Finally, *nonce* is a randomly generated value to uniquely identify an Interest packet.

Data Packet

A data packet is a response packet to the Interest packet and it consists of *content name*, *signature*, *signed info*, and *data* fields. *Content name* is the unique indicator for identifying the data packet. *Signature* is the authenticating information for the data packet, and it can be verified under the information *signed info* which includes the publisher's ID and key locator. A *data* field is the real data piece of a content.

2.1.2. CCN Forwarding Engine

A content requester sends an *I-Packet* containing the *content name* that he/she wants to get, and network nodes (forwarding engines, publisher, or other nodes), with the content matched to the *content name* in the Interest packet, send *D-Packet* through the reverse path, where the Interest comes from.

When a CCN-forwarding engine receives an Interest packet on some faces, it searches the CS-PIT-FIB sequence by using the content name in the Interest. If there is content in the Content Store (CS) matched to the Interest, the data packet containing the requested content will be sent out through the face that the Interest arrived on and the Interest will be discarded. If there is no matching content in the CS, the forwarding engine searches the Pending Interest Table (PIT). In other words, if there is an exact-matched PIT entry, the Interest's arrival face will be added to the corresponding entry's RequestingFaces list and the Interest will be discarded (the so-called Interest aggregation function). When a sata packet corresponding to this PIT entry arrives, the copy of the packet will be sent out through the faces recorded on the entry's RequestingFaces list and the entry will be removed in the PIT. If there is no matched entry in both CS and PIT (this means that this Interest is the first request to that content), the forwarding engine searches the Forwarding Interest Base (FIB). If there is a matching FIB entry with the longest-prefix

matching method, then the Interest will be sent out through faces on the Face list of the entry and a new PIT entry is created about the Interest and its arrival face. Finally, if there is no match for the Interest in CS, PIT and FIB, it is discarded.

2.1.3. Data Packet Authentication in (Naive) CCN

As a default data packet authentication, PARC implemented an MHT-based aggregate signing mechanism into the CCNx library, the CCN protocol's reference implementation. The CCNx library supports per-packet authentication and multi-packet authentication using MHT, depending on the security configuration [1–3]. Recently, Seo et al. [14] proposed an efficient two-layered data packet authentication mechanism in CCN.

The data packet can be authenticated with signature and signedinfo fields. Since all information required for verification is included in each data packet, any entity can perform the data packet verification, including forwarding engines. Note that forwarding engines do not verify the data packet in original CCN architecture even though they can. To use a signature scheme for verification, an architecture such as PKI is required to check the validity of a public key. Though some dedicated systems have been studied for CCN, a verifier is still needed to access the validation system to check the validity of given public key before verifying a signature using the public key [15]. Though the cost of public key validation was not seriously considered in measuring the cost of signature verification, it is not easy to ignore the cost, since at least one verification should be performed for the signature generated by a CA on the public key. If we have to use more complicates systems for checking the freshness of the public key by using a system such as CRL or OCSP, the cost of public key validation may take form a large portion of the total cost of the verification of the data packet.

2.2. Requirements

2.2.1. Security Model

Since generating and injecting invalid packets into the network is a very common and practical scenario, it is assumed that adversaries can perform one of the following attack scenarios. Before giving a formal definition, we recall some attack scenarios as follows.

Distribute Poisoned Data

The most harmful attack is the distribution of damaged data. Since a successful adversary can generate harmful but correctly verified data packets, requesters may receive the harmful data instead of the original data. If the poisoned data include simply garbage information, requesters fail to obtain the expected data. Though requesters can obtain the original data later by asking a reliable source, some services that need timely data transmission will be damaged. When the adversary succeeds in inserting a malicious code into the poisoned data packet, receivers can be damaged by the code. Since the damage caused by the code cannot be anticipated, we need a way to detect poisoned data for secure data distribution via CCN.

Fortunately, in CCN, the use of an authentication mechanism such as a digital signature scheme is considered to guarantee the integrity of data. Requesters can detect a corrupted data packet by verifying the authenticating message included in the data packet. Therefore, poisoned attacks can be easily countered by using a suitable authentication mechanism.

DDoS Attack

It is not easy to distribute a maliciously forged data packet in CCN due to the use of an authentication mechanism. However, it is still relatively easy to inject a meaningless data packet into CCN. Thus,

adversaries can mount a DDoS attack against CCN by generating a number of valid data packets and the corresponding number of request packets [15,16]. Though the robustness against DDoS attack is a merit of ICN, it is still possible to mount DDoS attacks by generating a number of data packets and the corresponding request packets if an adversary obtains the right to publish, which implies that (s)he can generate a valid data packet and publish a maliciously generated data packet into CCN. Note that an easy way to prevent the adversary from distributing maliciously made packets is maintaining a list of reliable publishers. In this case, only permitted publishers can generate the data packet, and so we should give up the convenience of being able to participate in publishing.

When a DDoS attack occurs, forwarding engines consume computing power to support the transmission of manipulated packets. Though CCN is designed to aggregate a number of packets into a sign packet, the adversary still can perform DDoS by generating a number of packets that cannot be aggregated. For example, a number of (different) data packets and the corresponding interest packets can be generated to be injected. Note that the packets cannot be aggregated since only interest packets for the same data packet can be aggregated.

Differently from an ordinary network, ICN is robuster against DDoS since the volume of manipulated packets influence the effect of DDoS and generating huge packets is also a burden to the adversary. However, ICN is still vulnerable to DDoS if the adversary can use sufficient resources to generate packets for attack.

Cache Consumption Attack

In ICN, forwarding engines maintain (relatively) large storage to cache data packets for faster data transmission. If the adversary generates and injects maliciously generated data packets into the network, the forwarding engines' cache would be filled with garbage data. Then, valid packets have less of a chance to be stored in the forwarding engines' storage, which influences the performance of forwarding, since the forwarding engine can reduce the round trip time by responding to an interest packet without asking other forwarding engines only if the required file is already stored in his storage. Cache consumption attack can be seen as a part of the DDoS attack, but different from the ordinary DDoS attack since the damage caused by cache consumption attacks lasts until the stored garbage data are removed from forwarding engines' caches.

The described attack scenarios can be countered if secure authentication technique is applied to CCN architecture appropriately and all forwarding engines verify every data packet under the authentication technique. To be sure, it is not easy to counter the case where an adversary obtains a legitimate right to publish and generate a number of data packets for malicious purposes. Therefore, the case will be excluded from our discussion. Except for this extraordinary case, if it is possible to figure out maliciously generated packets when they are inserted into CCN, all the above attacks can be countered. From now on, the attacks are referred to as forgery attacks, since, fundamentally, an adversary's goal in any attack is to generate a forged data packet. Based on this observation, the security of CCN against forgery attacks can be formally defined as the following.

Definition 1. *Let* Auth *be the authentication mechanism in CCN architecture. Let* $Adv(\mathrm{Auth}, \mathcal{A})$ *be the advantage of an adversary* $\mathcal{A}$ *in terms of forgery attacks against* Auth. *More concretely,* $Adv(\mathrm{Auth}, \mathcal{A})$ *can be understood as the probability of successful forgery generation in the sense that a forged data packet is injected into the CCN architecture by passing the verification procedure deployed in* Auth. *Then, the authentication mechanism is secure against forgery attacks if*

$$Adv = \max\{Adv(\mathrm{Auth}, \mathcal{A}) | \mathcal{A} \in \mathcal{ADV}\} < \varepsilon$$

where $\mathcal{ADV}$ is the set of all adversaries and ε is a negligible value. In other words, the authentication mechanism Auth *is secure against forgery attacks if any powerful adversary cannot generate a forged data packet which can pass the verification procedure in the mechanism.*

2.2.2. Functional Requirements

In addition to the security requirement, for security against forgery attacks, some functional requirements are required for effective packet verification in CCN. To date, the research has tried to improve the authentication procedure in CCN, as mentioned in the above. However, they focus on the performance in end-users or need some restrictions for supporting efficient packet verification in forwarding engines. Recall that the naive CCN architecture can support packet verification in all forwarding engines since the functionality is already built into each data packet. Unfortunately, when we focus on the verification in forwarding engines, existing technologies fail to give a better performance compared with the naive approach. To understand the above viewpoint, we recall some useful features that can be guaranteed by the naive approach:

Verification Independence

Since CCN is a general purpose architecture, it can cover various types of data, including small-sized data where the data can be transmitted in a single data packet. Thus, for CCN, it is desirable to give an authentication mechanism which can support efficient packet verification regardless of the size of data to be treated. To cover the requirement, we have to guarantee *verification independence* in the sense that each data packet can be verified without any additional information, such as a sibling path in MTH-based approaches.

On-the-Fly Data Packet Generation

MHT-based approaches are suitable for static data distribution. For real-time data, e.g., live-TV broadcasting, the data to be published cannot be fixed before generating and transmitting data packets, which means that some approaches, such as MHT-based methods, cannot be applied to such applications, since a hash-tree can be constructed only after all the data are given. Recall that the naive CCN is very suitable for on-the-fly packet generation, since each data packet can be generated without considering other data packets belonging to the same data.

Non-Deniability (or Traceability)

In many techniques, publishers are assumed to be honest entities. However, these days, more entities generate various data. For example, in YouTube, anyone can generate their own contents and distribute them via Website. Such changes suggest the need for new technologies that can counter the threats. Since CCN is devised to prepare the coming age, it is better to assume an arbitrary publisher instead of fixed reliable publishers. In this case, we face a new challenge of countering malicious publishers. One possible and easy remedy is to make list of verified publishers. However, the method cannot cover data transmission generated by unexpected publishers. A better choice is to make a list of malicious publishers and permit any publishers to publish their data. However, in this case, we need non-deniability (or traceability), since a malicious publisher should be identified. In naive CCN, the use of a public-key-based signature scheme is considered as a mandatory component, which implies that the naive CCN can support the functionality due to the non-deniability of the underlying signature scheme.

Setup-Free Construction

One significant goal of CCN is mobility in many aspects. However, many existing works weaken the feature to improve the performance of authentication. For example, HMAC-based methods and the Token-based method are designed by assuming some pre-shared secret information, or by parties needing to maintain common information which is necessary for being synchronized. If we need any setup procedure for authentication, it could influence the mobility of CCN, since any entity who wants to participate in the communication should set up according to the underlying system. For example, in the HMAC-based system, each user should shares common secret information with the publisher before participating in the system. One of the demerits of state-based construction is that parties need to maintain secret information and the existence of secret information in many points causes various security threats. Hence, it is obvious that the setup-freeness is one of the important requirements of CCN.

Recall that any attacks considered in Section 2.2.1 can be countered by verifying the validity of data packets, and we can achieve the security goal by using public-key-based signature schemes. However, to achieve all the above requirements is not easy, since the use of heavy cryptographic primitive caused by the use of signature schemes can be a hurdle to overcome. For example, if we use an MHT-based approach, we can support the above requirements efficiently except the second requirement. If we adopt the naive authentication mechanism in CCN and let all forwarding engines to verify all data packets, we can achieve all the above requirements. However, the naive approach requires heavy computation in forwarding engines. Though the efficiency of verification in forwarding engines is not explicitly included in the list of requirements, it is very important in the authenticating mechanism for its practicality. Hence, the main goal of the paper is to design an authentication mechanism which can achieve all requirements with very low cost.

3. Proposed Overlaid Authentication Mechanism

In this section, a new architecture named *TAuth* will be described, which supports secure and efficient routing in a content-centric network (CCN).

3.1. Basic Idea

To accelerate the authenticated packet forwarding in CCN, network connections are divided into two types according to the reliability of connections. The connections between a forwarding engine and an outsider, such as a requester or a data publisher, cannot support strong reliability since the forwarding engine cannot trust the outsider and vice versa. On the other hands, the connection between two forwarding engines can support stronger trust since there is a fixed number of forwarding engines in the network, and it is possible to maintain the list of forwarding engines and the corresponding information, such as the list of their public keys.

Note that a data packet put into the network passes through a number of forwarding engines until it reaches a requester. When the data is first given to a forwarding engine, it verifies the given packet using attached authenticating information such as signature and signed information. If the packet is verified to be valid, the forwarding engine sends the packet to adjacent forwarding engines. Then, the forwarding engines do the same operations until the packet reaches the requester. Note that most of the data transmission and verification are executed by many forwarding engines who can trust themselves more than outsiders. Based on the observation, it is possible to design an efficient authentication mechanism by providing an overlay for connections between forwarding engines. Due to the stronger reliability between forwarding engines, the cost for authentication is reduced in many viewpoints. Finally, note that, the main idea of this work is to divide network links into two levels according to the reliability between

nodes, and use less-intensive and efficient authentication for reliable nodes, which form a large part of data transmission.

3.2. Components

3.2.1. Signature Scheme for Content Publishers

The verification algorithm optionally requires the message as an input. For example, most well-known signature schemes such as DSA require the signed message as an input. However, the value is omitted from the input of the function, since it does not influence the readability of the technique and some signature schemes such as RSA do not require the signed message as an input. We assume that each content publisher Pub_i may use a different signature scheme, and so we let $\{\mathsf{Sig}_i(s_i, m), \mathsf{Ver}_i(v_i, \sigma)\}$ be the publisher's signing and verification algorithms where:

- s_i is the private signing key;
- v_i is the corresponding public verification key;
- m is a message to be signed;
- σ is a signature generated by the signature scheme.

3.2.2. Signature Scheme for Forwarding Engine

Differently from the signature for publishers, it is possible to regulate forwarding engines' signature schemes. To distinguish forwarding engines' algorithms from publishers' algorithms, different notations are used as follows. Let $\{\mathsf{SIG}(sk_i, m), \mathsf{VER}(vk_i, \sigma)\}$ be the signature scheme used by forwarding engines.

3.2.3. Public Key Verification

For authenticating data packets, the signature scheme is used as a building block. In general, the cost of signing and verification is considered to measure the efficiency of a signature-based authenticating technique, but the cost of examining the freshness of a used public key is frequently missed. To prove the freshness of a key, we need to access to an archive, such as the certificate revocation list (CRL) in public key infrastructure (PKI), which maintains the most recent information regarding the freshness of public keys. Each user should interact with the archive to check the freshness of a public key before using it to verify some information, including data packets. Let $\mathsf{KVer}(v_i)$ be the public key verification function which returns a binary response according to the freshness of the input public key. For the ordinary public-key-based scheme, the function can be seen as an interactive protocol. For some schemes, such as ID-based schemes and attribute-based schemes, the function can be executed locally. In this case, the cost of communication for performing interactive protocol can be reduced, but we face other problems instead. In the schemes, each public key is frequently updated to support the freshness without proving it via interactive protocol.. For CCN, dedicated systems have been considered instead of the ordinary PKI. However, even though the systems are adopted in CCN, we still need additional costs for checking the freshness of public keys.

Most operations for CCN data transmissions are executed in forwarding engines, and the main difference between our architecture and other authentication mechanisms for CCN is the way of dealing authenticating messages between forwarding engines. Recall that forwarding engines can trust each other with higher reliability than other entities in CCN, and we can improve the performance of data transmission by reducing the cost of packet verification based on this assumption. In our construction, forwarding engines maintain $List_R$, the list of forwarding engines and their public keys, i.e., $List_R = \{(rid_i, vk_i) | i = 1, \ldots, n_R\}$ where n_R is the number of forwarding engines. Instead of relying on a system

such as PKI, forwarding engines in our construction use the list to check the correctness of public keys. Note that the list is used not for all public keys, but for forwarding engines.

How to construct the list is an important issue in this approach. Fortunately, we already have a number of solutions for this problem. As a simple solution, the network service provider *NSP* can intervene in the distribution of the list $List_R$. Since *NSP* knows the network topology and the list of forwarding engines, it is relatively easy to make the list. Even if *NSP* does not construct the list on behalf of the forwarding engines, each forwarding engine can construct the list by adding a new public key when it receives a forwarding engine-generated packet which is generated by a new forwarding engine.

3.3. Authenticated Data Generation and Verification

In CCN, the three entities work as follows.

3.3.1. Publisher Side

The role of a publisher Pub_i is to generate data packets and publish them for its own data. The publisher Pub_i generates and publishes its own data as follows. To publish a file F of which file ID is *fid*, the data publisher splits the file into n blocks as

$$F = m_1||m_2||\cdots||m_n.$$

Then, Pub_i runs the data packet generation function

$$D\text{-}Packet(pid_i, F) = \{DP_{i,1}, DP_{i,2}, \ldots, DP_{i,n}\}$$

where $DP_{i,j}$ is the data packet for each block m_j. Specifically, each data packet is defined as

$$DP_{i,j} = c\text{-}name_j||\sigma_{i,j}||auth\text{-}info_i||m_j$$

where:

- *c-name* is the content name of the file F;
- $c\text{-}name_j = c\text{-}name||j$;
- $M_j = c\text{-}name_j||auth\text{-}info_i||m_j$;
- $\sigma_{i,j} = \mathsf{Sig}_i(s_i, M_j)$;
- $auth\text{-}info_i = \{pid_i, \mathsf{Ver}_i, v_i\}$.

Finally, the publisher Pub_i transmits the data packets into networks. In this phase, the publisher's data packet will be firstly sent to the closest forwarding engine. The publisher may send the data packet to a set of close forwarding engines, but the prosecure is stated as in the above to simplify the description of the proposed scheme, since the feature will be used to improve the performance of authentication mechanism in CCN.

The pseudo-code for two roles, generation of authenticated content packet and transmission of authenticated data packet, are given in Algorithms 1 and 2. Note that both algorithms are included to aid the reader's understanding. The publisher can generate and publish its own data as described in the above, and specific procedures are written as pseudo-code in Algorithms 1 and 2.

Algorithm 1 Authenticated Content Generation at Content Publisher (DPG)

Require: pid_i, Content $m = (m_1||m_2||\cdots||m_n)$, content name *c-name*, signing key s_i.
Ensure: $D\text{-}Packet = (DP_{i,1}, DP_{i,2}, \ldots, DP_{i,n})$.
1: **for** j from 1 downto n **do**
2: Construct $c\text{-}name_j$ by concatenating *c-name* and packet index j
3: Construct M_j by concatenating $c\text{-}name_j$, $auth\text{-}info_i$, and m_j
4: Construct $\sigma_{i,j} = \mathsf{Sig}_i(s_i, M_j)$
5: Construct $auth\text{-}info_i \leftarrow pid_i, Ver_i, v_i$
6: $DP_{i,j} \leftarrow (c\text{-}name_j||\sigma_{i,j}||auth\text{-}info_i||m_j)$
7: **end for**
8: **return** Authenticated data packets as $D\text{-}Packet = (DP_{i,1}, DP_{i,2}, \ldots, DP_{i,n})$

Algorithm 2 Transmitting Authenticated Data Packets at Content Publisher Pub_i

Require: Content $F = (m_1||m_2||\cdots||m_n)$, signing key s_i.
Ensure: Send authenticated data packets.
1: /* Generating signed data packets */
2: $D\text{-}Packet = (DP_{i,1}, DP_{i,2}, \ldots, DP_{i,n}) \leftarrow \mathtt{DPG}(F, s_i)$
3: **while** Listen Interest packets from other nodes **do**
4: Receive Interest *I-Packet*
5: Get the content name *c-name* from the received *I-Packet*
6: **if** there is a match in $DP_{i,v}$ **then**
7: Send $DP_{i,v}$ to the network
8: **end if**
9: **end while**

3.3.2. Consumer Side

To obtain the file F, an requester runs the Interest packet generation function as

$$I\text{-}Packet(uid, fid) = c\text{-}name||selector||nonce$$

where:

- *uid* is the user's ID;
- *selector* is a set of rules for obtaining the target file;
- *nonce* is a random nonce for uniquely identifying messages.

Typically, the requester sequentially sends the Interest packet for the first data packet to the n-th data packet. When the requester receives the data packet ($DP_{i,j}$ for $1 \leq j \leq n$) corresponding to the Interest packet, he/she runs the verification function for the publisher Pub_i by testing the following condition

$$\mathsf{Ver}_i(v_i, \sigma_{i,j}) = \mathsf{TRUE}.$$

If all data packets are verified to be valid, the user accepts the received data packets and reconstructs the file F.

3.3.3. Forwarding Engine

In our technique, forwarding engines perform two types of operation: the verification of publisher's data packet and the generation of authenticating information for higher-layer. Specific operations for the two types will be described in the following:

Data Publisher $\rightarrow$ Forwarding Engine

When a forwarding engine receives a data packet $DP_{i,j} = c\text{-}name_j||\sigma_{i,j}||auth\text{-}info_i||m_j$ from a data publisher Pub_i, the forwarding engine verifies the packet by testing the following condition

$$\mathsf{Ver}_i(v_i, \sigma_{i,j}) = \mathsf{TRUE}.$$

If the data packet is verified as valid one, the forwarding engine who firstly receives and verifies the packet generates a new authenticating message as

$$\sigma_{i,j,k} = \mathsf{SIG}(sk_k, DP_{i,j}).$$

Then, the forwarding engine constructs a new data packet as

$$DP_{i,j,k} = c\text{-}name_j||\{\sigma_{i,j}, \sigma_{i,j,k}\}||\{auth\text{-}info_i, vk_k\}||m_j.$$

Forwarding Engine $\rightarrow$ Forwarding Engine

When a forwarding engine receives a data packet $DP_{i,j,k}$ from another forwarding engine, the forwarding engine checks if the public key used for generating the packet belongs to $List_R$, i.e., verifies whether the packet is re-signed by a valid forwarding engine or not. If so, the forwarding engine verifies the given packet by testing the following condition

$$\mathsf{VER}(vk_k, \sigma_{i,j,k}) = \mathsf{TRUE}.$$

If the verification holds, the forwarding engine sends the data packet to the next forwarding engine. Recall that the main merit of our construction is the use of the same signature scheme between forwarding engines. Since data publishers' data packets are encapsulated by forwarding engines who receive the packets the first time, we can see that all data packets can be verified under the same verification equation regardless of the signature scheme used by data publishers.

Forwarding Engine $\rightarrow$ Data Requester

When a forwarding engine sends a data packet to a requester, the forwarding engine recovers the original data packet $DP_{i,j}$ from the modified data packet $DP_{i,j,k}$, and gives $DP_{i,j}$ to the requester. We still can successfully finish the delivery of data without recovering the original data packet. However, we need this step to protect the location privacy of the data publisher. If the forwarding engine gives the modified packet instead of the original packet, the requester can guess the physical location of the publisher by searching the location of the forwarding engine, whose public verification key is vk_k.

4. Security Analysis

The proposed architecture is secure if an adversary cannot generate and inject a forged data packet into CCN, and the security feature can be proved by showing that the proposed architecture is robust against forgery attacks. Here, the security of the proposed architecture will be proved as follows.

Theorem 1. *Let $\mathcal{A}$ be an adversary who tries to break the security of the proposed* TAuth *architecture with q_{PC} publisher corruption queries and q_{RC} forwarding engine corruption queries. It is assumed that the adversary is permitted to, at most, $n_R/2$ forwarding engine corruption queries and $n_P/2$ publisher corruption queries. Then, the security of the proposed* TAuth *architecture against the adversary $\mathcal{A}$ who tries to break the authentication architecture in terms of forgery attacks is guaranteed by the unforgeability of underlying signature schemes.*

Proof. To prove the security of the proposed architecture, we will show that a successful attack induces the insecurity of an underlying signature scheme. Let $\mathcal{A}_{TAuth}$ be an adversarial algorithm which breaks the security of *TAuth* by generating invalid data packets. Our strategy is to construct an algorithm $\mathcal{A}_{Sig}$ which breaks the security of an underlying signature scheme by using $\mathcal{A}_{TAuth}$ as a sub-algorithm. Let $\{\mathbf{Sig}(\cdot,\cdot), \mathbf{Ver}(\cdot,\cdot)\}, vk$ be a set of algorithms and public key given to $\mathcal{A}_{Sig}$ as a challenge. Note that $\mathcal{A}_{Sig}$ can access to the signing oracle $\mathcal{O}_{\mathsf{S}}(\cdot)$, which returns a signature σ for a message m such that $\mathbf{Sig}(sk, m) = \sigma$ where sk is the private key for vk.

Recall that the goal of $\mathcal{A}_{TAuth}$ is to inject the manipulated data packet which can pass the pre-defined verification procedure in ICN. Hence, the adversary may return one of the following valid data packets as a result of its attack:

- *Type-0* forgery $DP_{i,j}$: Original data packet generated by publisher Pub_i;
- *Type-1* forgery $DP_{i,j,k}$: Data packet for higher layer, which is re-generated by forwarding engine Rou_k.

For each type, a different strategy is required to use $\mathcal{A}_{TAuth}$, since we have to resolve the given challenge to the attack environment for $\mathcal{A}_{TAuth}$ so that the output of the algorithm is useful for the given problem. To guess the type, before beginning the game, $\mathcal{A}_{Sig}$ chooses a bit $b \in_R \{0, 1\}$. According to the bit, $\mathcal{A}_{Sig}$ simulates an attack environment and oracle queries for $\mathcal{A}_{TAuth}$ as follows.

Simulation for *Type-0 Forgery*: Since we assume that the forgery may belong to *Type-0*, we use the adversarial algorithm to break the security of a signature scheme $\{\mathsf{Sig}(\cdot,\cdot), \mathsf{Ver}(\cdot,\cdot)\}$ of which the public verification key is v. Note that we can access $\mathcal{O}_{\mathsf{S}}(\cdot)$, the signing oracle for the target signature scheme, which returns a signature for given a message. Note that in ICN, the adversary $\mathcal{A}$ can ask *data packet generation query*. However, we describe it as a signing query, since the core procedure of data packet generation is the signature generation. Hence, we will describe *signing query* instead of *data packet generation query*. To respond to all queries given by $\mathcal{A}$, we do the following:

- *Signing Queries.* In the beginning of the simulation, $\mathcal{A}_{\mathsf{Sig}}$ guesses a publisher Pub^* who will be attacked by the adversary $\mathcal{A}$. For a publisher $Pub_i (\neq Pub^*)$, $\mathcal{A}_{\mathsf{Sig}}$ chooses a signature scheme $\{\mathsf{Sig}_i(\cdot,\cdot), \mathsf{Ver}_i(\cdot,\cdot)\}$ and sets $\{s_i, v_i\}$ as the signing/verification key pair for the scheme. $\mathcal{A}_{\mathsf{Sig}}$ maintains List_{Pub} the list of i, $\{\mathsf{Sig}_i(\cdot,\cdot), \mathsf{Ver}_i(\cdot,\cdot)\}$, and $\{s_i, v_i\}$). When $\mathcal{A}$ asks a signature of $Pub_i (\neq Pub^*)$ on M, $\mathcal{A}_{\mathsf{Sig}}$ responses to the query by generating a valid signature as $\sigma_{i,M} = \mathsf{Sig}_i(s_i, M)$. If the query is made for the target publisher Pub^* on M, $\mathcal{A}_{\mathsf{Sig}}$ uses the signing oracle to obtain $\sigma_{*,M} = \mathcal{O}_{\mathsf{S}}(M)$ and gives it to $\mathcal{A}$.

  For each forwarding engine, $\mathcal{A}_{\mathsf{Sig}}$ chooses a signature scheme and the corresponding key pairs. Let $\{\mathsf{SIG}_k(\cdot,\cdot), \mathsf{VER}_k(\cdot,\cdot)\}$ and $\{S_k, V_k\}$ be the signature scheme and key pairs of the forwarding engine Rou_k. $\mathcal{A}_{\mathsf{Sig}}$ also maintains List_{Rou} the list of k, $\{\mathsf{SIG}_k(\cdot,\cdot), \mathsf{VER}_k(\cdot,\cdot)\}$, and $\{S_k, V_k\}$). For a signing query on M, $\mathcal{A}_{\mathsf{Sig}}$ generates a valid signature as $\sigma_{i,M,k} = \mathsf{SIG}_k(S_k, M)$;
- *Publisher Corrupt Queries.* In the adversary model, we assumed that a set of publishers can collude to break the proposed architecture, which implies that $\mathcal{A}$ can ask for the private key of a publisher Pub_i. Here, we assume that the queried key exists in List_{Pub}, and the assumption is reasonable since $\mathcal{A}_{\mathsf{Sig}}$ can generate the publisher's key information, as in the above, before response to the corrupt query.

For the corrupt query, $\mathcal{A}_{\mathsf{Sig}}$ retrieves i, $\{\mathsf{Sig}_i(\cdot,\cdot), \mathsf{Ver}_i(\cdot,\cdot)\}$, and $\{s_i, v_i\}$ from List_{Pub} and gives s_i to $\mathcal{A}$. If the queried publisher is Pub^*, $\mathcal{A}_{\mathsf{Sig}}$ stops the simulation;

- *Forwarding Engine Corrupt Queries.* In this case, it is assumed that $\mathcal{A}$ can ask for the private key of a forwarding engine Rou_i. For *Type-0* forgery, $\mathcal{A}_{\mathsf{Sig}}$ retrieves i, $\{\mathsf{SIG}_i(\cdot,\cdot), \mathsf{VER}_i(\cdot,\cdot)\}$, and $\{S_i, V_i\}$ from List_{Rou} and gives S_i to $\mathcal{A}$.

Recall that the adversary $\mathcal{A}$'s goal is to generate a valid data packet without the private information of the corresponding publisher. Let $DP_{*,j}$ be the forged data packet with respect to a message M^*. Then, $\mathcal{A}_{\mathsf{Sig}}$ extracts the authenticating message σ_{*,M^*} from the data packet. If σ_{*,M^*} is a valid signature of Pub_*, $\mathcal{A}_{\mathsf{Sig}}$ returns $\{\sigma_{*,M^*}, M^*\}$. Otherwise, $\mathcal{A}_{\mathsf{Sig}}$ stops the simulation.

Simulation for *Type-1 Forgery*: In this case, we use the adversarial algorithm to break the security of a signature scheme $\{\mathsf{SIG}(\cdot,\cdot), \mathsf{VER}(\cdot,\cdot)\}$ of which the public verification key is V. Since our goal is to break the security of the signature scheme, we can access the signing oracle $\mathcal{O}_{\mathsf{S}}(\cdot)$, which returns a signature for a given message. Similar to the simulation for *type-0 forgery*, we respond to all queries given by $\mathcal{A}$ as follows:

- *Signing Queries.* For each publisher Pub_i, $\mathcal{A}_{\mathsf{Sig}}$ chooses a signature scheme and the corresponding key pairs. Let $\{\mathsf{Sig}_i(\cdot,\cdot), \mathsf{Ver}_i(\cdot,\cdot)\}$ and $\{s_i, v_i\}$ be the signature scheme and key pairs of the publisher Pub_i. $\mathcal{A}_{\mathsf{Sig}}$ also maintains List_{Pub} the list of i, $\{\mathsf{Sig}_i(\cdot,\cdot), \mathsf{Ver}_i(\cdot,\cdot)\}$, and $\{s_i, v_i\}$). For a signing query on M, $\mathcal{A}_{\mathsf{Sig}}$ generates a valid signature as $\sigma_{i,M} = \mathsf{Sig}_i(s_k, M)$ and gives it to $\mathcal{A}$.

  Differently from the simulation for *type-1 forgery*, $\mathcal{A}_{\mathsf{Sig}}$ guesses a forwarding engine Rou^* who will be attacked by the adversary $\mathcal{A}$. For a forwarding engine $Rou_k(\neq Rou^*)$, $\mathcal{A}_{\mathsf{Sig}}$ chooses a signature scheme $\{\mathsf{SIG}_k(\cdot,\cdot), \mathsf{VER}_k(\cdot,\cdot)\}$ and a sets $\{S_k, V_k\}$ as the signing/verification key pair for the scheme. $\mathcal{A}_{\mathsf{Sig}}$ maintains List_{Rou} the list of k, $\{\mathsf{SIG}_k(\cdot,\cdot), \mathsf{VER}_k(\cdot,\cdot)\}$, and $\{S_k, V_k\}$). When $\mathcal{A}$ asks a signature of $Rou_k(\neq Rou^*)$ on M, $\mathcal{A}_{\mathsf{Sig}}$ responses to the query by generating a valid signature as $\sigma_{i,M,k} = \mathsf{SIG}_k(S_k, M)$. If the query is made for the target publisher Rou^* on M, $\mathcal{A}_{\mathsf{Sig}}$ uses the signing oracle to obtain $\sigma_{*,M,k} = \mathcal{O}_{\mathsf{S}}(M)$ and gives it to $\mathcal{A}$;
- *Publisher Corrupt Queries.* When $\mathcal{A}$ asks the private key of a publisher Pub_i, $\mathcal{A}_{\mathsf{Sig}}$ retrieves i, $\{\mathsf{Sig}_i(\cdot,\cdot), \mathsf{Ver}_i(\cdot,\cdot)\}$, and $\{s_i, v_i\}$ from List_{Pub} and gives s_i to $\mathcal{A}$;
- *Forwarding Engine Corrupt Queries.* When $\mathcal{A}$ asks the private key of the forwarding engine $Rou_k(\neq Rou^*)$, $\mathcal{A}_{\mathsf{Sig}}$ retrieves k, $\{\mathsf{SIG}_K(\cdot,\cdot), \mathsf{VER}_k(\cdot,\cdot)\}$, and $\{S_k, V_k\}$ from List_{Rou} and gives S_k to $\mathcal{A}$. If the queried forwarding engine is Rou^*, $\mathcal{A}_{\mathsf{Sig}}$ stops the simulation.

From now, we will measure the success probability of the adversary $\mathcal{A}_{\mathsf{Sig}}$. Let $Adv^{\mathsf{EUF}}_{\mathcal{A}_{\mathsf{Sig}}}$ be the advantage of $\mathcal{A}_{\mathsf{Sig}}$ regarding the existential unforgeability of the given challenge signature scheme. We say that a signature returned by $\mathcal{A}_{TAuth}$ is valid forgery, if and only if, the signature can be verified under the challenge parameters and is not generated by an oracle call. Then, the advantage of $\mathcal{A}_{\mathsf{Sig}}$ can be expressed as

$$Adv^{\mathsf{EUF}}_{\mathcal{A}_{\mathsf{Sig}}} = \mathsf{Pr}[\mathcal{A}_{TAuth} \text{ returns a valid signature.}$$

Let $\mathsf{E}_{\alpha,\beta}$ be the event where $\mathcal{A}_{TAuth}$ returns a *Type-*β forgery after the simulation while $\mathcal{A}_{Sig}$ chooses α at the beginning of the simulation and successfully simulates attack environment for $\mathcal{A}_{TAuth}$ according to the bit α. Then, $Adv^{\mathsf{EUF}}_{\mathcal{A}_{\mathsf{Sig}}}$ can be rewritten as

$$Adv^{\mathsf{EUF}}_{\mathcal{A}_{\mathsf{Sig}}} = \sum_{\alpha,\beta\in\{0,1\}} \mathsf{Pr}[\mathsf{E}_{\alpha,\beta}] \cdot \mathsf{Pr}[\mathsf{E}_{valid}|\mathsf{E}_{\alpha,\beta}]$$

where E_{valid} is the event that the returned signature is valid in $\mathcal{A}_{\mathsf{Sig}}$'s viewpoint. In $\mathsf{E}_{0,1}$ and $\mathsf{E}_{1,0}$, the algorithm $\mathcal{A}_{Sig}$ fails to obtain a valid forgery using $\mathcal{A}_{TAuth}$ since the given challenge is not correctly

resolved in the simulation, which means that $\mathcal{A}_{TAuth}$'s output is invalid in two events. Then, we can simplify the advantage of $\mathcal{A}_{Sig}$ as

$$Adv^{\mathsf{EUF}}_{\mathcal{A}_{\mathsf{Sig}}} = \mathsf{Pr}[\mathsf{E}_{0,0}] \cdot \mathsf{Pr}[\mathsf{E}_{valid}|\mathsf{E}_{0,0}] + \mathsf{Pr}[\mathsf{E}_{1,1}] \cdot \mathsf{Pr}[\mathsf{E}_{valid}|\mathsf{E}_{1,1}].$$

In $\mathsf{E}_{0,0}$ and $\mathsf{E}_{0,0}$, $\mathcal{A}_{Sig}$ constructs a valid attack environment for $\mathcal{A}_{TAuth}$, and thus $\mathsf{Pr}[\mathsf{E}_{valid}|\mathsf{E}_{b,b}] = \varepsilon$ for all $b = 0, 1$ where ε is the advantage of $\mathcal{A}_{TAuth}$. For $b = 0, 1$, the event $\mathsf{E}_{b,b}$ implies two conditions: (1) $\mathcal{A}_{Sig}$ correctly guesses the type of $\mathcal{A}_{TAuth}$'s forgery and (2) $\mathcal{A}_{Sig}$'s simulation is valid for $\mathcal{A}_{TAuth}$. Since the two conditions are statistically independent, we have

$$\mathsf{Pr}[\mathsf{E}_{b,b}] = \mathsf{Pr}[\mathsf{E}^1_{b,b}] \cdot \mathsf{Pr}[\mathsf{E}^2_{b,b}]$$

where $\mathsf{E}^1_{b,b}$ and $\mathsf{E}^2_{b,b}$ are the events that the conditions 1 and 2 hold, respectively. Let p be the probability that $\mathcal{A}_{TAuth}$ returns *Type-0* forgery. Then

$$\mathsf{Pr}[\mathsf{E}^1_{0,0}] = \frac{p}{2} \text{ and } \mathsf{Pr}[\mathsf{E}^1_{1,1}] = \frac{1-p}{2}.$$

We then measure the upper-bound of the probability $\mathsf{Pr}[\mathsf{E}^2_{0,0}]$ and $\mathsf{Pr}[\mathsf{E}^2_{1,1}]$. As seen in the simulation for *Type-0* forgery, $\mathcal{A}_{\mathsf{Sig}}$ gives up the simulation when $\mathcal{A}_{TAuth}$ made a *publisher corrupt query* for Pub^* or $\mathcal{A}_{TAuth}$ does not return a signature of Pub^*. Hence, the probability of success simulation can be interpreted as

$$\begin{aligned}
\mathsf{Pr}[\mathsf{E}^2_{0,0}] &= \Big(1 - \frac{1}{n_P}\Big)\Big(1 - \frac{1}{n_P\text{-}1}\Big) \cdots \Big(1 - \frac{1}{n_P\text{-}q_{PC}\text{+}1}\Big)\frac{1}{n_P} \\
&\geq \Big(1 - \frac{1}{n_P\text{-}q_{PC}\text{+}1}\Big)^{q_{PC}}\frac{1}{n_P}
\end{aligned}$$

where n_P is the number of publishers and q_{PC} is the number of publisher corruption queries. Similarly, the probability of success simulation for *Type-1* forgery can be expressed as

$$\mathsf{Pr}[\mathsf{E}^2_{1,1}] \geq \Big(1 - \frac{1}{n_R\text{-}q_{RC}\text{+}1}\Big)^{q_{RC}}\frac{1}{n_R}$$

where n_R is the number of forwarding engines and q_{RC} is the number of forwarding engine corruption queries. Note that $(1 - 1/x)^x \approx 1/e$ for large x. Hence, we have

$$\begin{aligned}
\mathsf{Pr}[\mathsf{E}^2_{0,0}] &\geq \Big(1 - \frac{1}{n_P\text{-}q_{PC}\text{+}1}\Big)^{q_{PC}}\frac{1}{n_P} \\
&= \left\{\Big(1 - \frac{1}{n_P\text{-}q_{PC}\text{+}1}\Big)^{n_P\text{-}q_{PC}\text{+}1}\right\}^{\frac{q_{PC}}{n_P\text{-}q_{PC}\text{+}1}}\frac{1}{n_P} \\
&\approx \frac{1}{e^{\frac{q_{PC}}{n_P\text{-}q_{PC}\text{+}1}}n_P}.
\end{aligned}$$

Since we assumed that $\mathcal{A}_{TAuth}$ makes, at most, $n_P/2$ publisher corruption queries, we can simplify the above inequality as

$$\mathsf{Pr}[\mathsf{E}^2_{0,0}] \geq \frac{1}{en_P}.$$

Similarly, we also have

$$\mathsf{Pr}[\mathsf{E}^2_{1,1}] \geq \frac{1}{en_R}.$$

Then, the advantage of $\mathcal{A}_{\mathsf{Sig}}$ can be estimated as

$$\begin{aligned} Adv^{\mathsf{EUF}}_{\mathcal{A}_{\mathsf{Sig}}} &= (\Pr[\mathsf{E}_{0,0}] + \Pr[\mathsf{E}_{1,1}]) \cdot \varepsilon \\ &\geq \left(\frac{p}{2en_P} + \frac{1-p}{2en_R}\right) \cdot \varepsilon \\ &= \left\{\frac{1}{2en_R} - \frac{p}{2e}\left(\frac{1}{n_R} - \frac{1}{n_P}\right)\right\} \cdot \varepsilon \\ &\geq \frac{1}{2en_R}\varepsilon \end{aligned}$$

where ε is the advantage of $\mathcal{A}_{TAuth}$. The last inequality comes from the fact that $n_R < n_P$. If $\mathcal{A}_{TAuth}$ has non-negligible advantage ε, we can construct $\mathcal{A}_{\mathsf{Sig}}$ which can break the security of an underlying signature scheme with non-negligible advantage $\varepsilon/2en_R$. Hence, we can conclude that the security of *TAuth* is guaranteed by the security of the underlying signature schemes. □

5. Instantiation

The proposed *TAuth* is a very flexible structure which can be implemented using any signature scheme. Since the underlying signature scheme influences the performance and security features of the architecture, it is very important to choose a suitable signature scheme to construct a concrete *TAuth* architecture. In CCN, a signature resolved in a data packet is verified several times according to the forwarding path. Therefore, for faster forwarding, the efficiency of verification is more important than the efficiency of signing. From this perspective, a concrete description for *TAuth* will be given.

5.1. Data Publisher

Recall that signature schemes used by publishers are not described as in Section 3, since publishers can use any signature scheme instead of using a fixed scheme. Let $\{\mathsf{Sig}_i(\cdot,\cdot), \mathsf{Ver}_i(\cdot,\cdot)\}$ be the publisher Pub_i's signing and verification algorithms and $\{s_i, v_i\}$ be the publisher's private signing key and the corresponding public verification key. A publisher Pub_i generates data packets for a file

$$F = m_1||m_2||\cdots||m_n$$

of which file ID is *fid* as

$$\textit{D-Packet}(pid_i, F) = \{DP_{i,1}, DP_{i,2}, \ldots, DP_{i,n}\}$$

where

$$DP_{i,j} = \textit{c-name}_j||\sigma_{i,j}||\textit{auth-info}_i||m_j,$$

c-name is the content name of the file F, $\textit{c-name}_j = \textit{c-name}||j$, $\sigma_{i,j} = \mathsf{Sig}_i(s_i, M_j)$, $M_j = \textit{c-name}_j||\textit{auth-info}_i||m_j$, and $\textit{auth-info}_i = \{pid_i, \mathsf{Ver}_i, v_i\}$. Then, the publisher Pub_i transmits data packets into networks. Refer to Section 3 for detail.

5.2. Data Consumer

Identical to the description in Section 3. Refer to the section for detail.

5.2.1. Forwarding Engine

Since forwarding engines have the RSA signature scheme in common, each forwarding engine Rou_k has his RSA public key (n_k, e_k) and the corresponding private key d_k, such that $e_k \cdot d_k = 1 \mod \phi(n_k)$.

Since the RSA signature is a PKI-based scheme, we need a way to verify the freshness of a public key. Due to the leveled reliability considered in this paper, we can lighten the cost of freshness test for public key between forwarding engines by maintaining a list $List_R$ which includes $\{rid_k, (n_k, e_k)\}$, the list of legitimate forwarding engines and their public keys. Instead of relying on PKI, forwarding engines can use other forwarding engines' public keys by sharing the list and checking whether a public key is included in the list.

Data Publisher $\rightarrow$ Forwarding Engine

When a data publisher Pub_i sends a data packet $DP_{i,j}$ a forwarding engine Rou_k, the forwarding engine verifies the given packet by testing $\mathsf{Ver}_i(v_i, \sigma_{i,j}) = \mathsf{TRUE}$. If the test holds, the forwarding engine computes a new signature for the data packet as

$$\sigma_k = H(DP_{i,j})^{d_k} \mod n_k.$$

Then, the forwarding engine constructs a new data packet as

$$DP_{i,j,k} = c\text{-}name_j || \sigma_{i,j,k} || auth\text{-}info_{i,k} || m_j$$

where $\sigma_{i,j,k} = \sigma_{i,j} || \sigma_k$ and $auth\text{-}info_{i,k} = auth\text{-}info_i || rid_k$.

Forwarding Engine $\rightarrow$ Forwarding Engine

In this case, the sender simply gives a data packet to the receiver. When a forwarding engine receives $DP_{i,j,k}$ from another forwarding engine, the receiver searches (n_k, e_k) from $List_R$ using rid_k and verifies the given packet by testing the equality

$$H(DP_{i,j}) = \sigma_k^{e_k} \mod n_k.$$

If the condition holds, the receiver sends the packet to the next forwarding engine, and otherwise drops the packet from the network.

Forwarding Engine $\rightarrow$ Data Consumer

After verifying a data packet $DP_{i,j,k}$, the sender forwarding engine truncates $\{\sigma_{i,j,k}, auth\text{-}info_{i,k}\}$ to obtain $\{\sigma_{i,j}, auth\text{-}info_i\}$, reconstructs the original data packet $DP_{i,j}$, and sends $DP_{i,j}$ to the data consumer.

The security of *RSA-TAuth* against forgery attacks can be reduced to the unforgeability of the RSA signature or the publisher's signature scheme, since adversaries should generate a valid signature of a forwarding engine or the publisher by Theorem 1.

6. Comparisons

In this section, the proposed mechanism is compared with other existing data packet authentication mechanisms in CCN with respect to the requirements described in Section 2.2.2 and verification cost.

6.1. Requirement Comparison

In Section 2.2.2, requirements for a secure and efficient data packet authentication mechanism in CCN are introduced. They are *verification independence*, *on-the-fly data packet generation*, *non-deniability*, and *setup-free construction*. Table 1 compares the existing data packet authentication mechanisms, including the proposed mechanism with respect to the defined requirements. It is assumed that the digital signature mechanism such as RSA-PSS is used for primitive signing, and every router on the routing path

verifies the received data packets through PKI. Since the Naive CCN signs every packet belonging to a requested content, it satisfies all requirements. Lightweight Integrity Verification architecture (LIVE) provides an efficient data packet integrity mechanism and allows a content publisher to control access to contents stored in remote CCN network nodes [11]. In LIVE, a content publisher distributes 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. Since LIVE requires the distribution of secure tokens and one-time signature (OTS) for efficiency, it fails to provide *on-the-fly data packet generation*, *non-deniability*, and *setup-free construction*. CCNx from [1–3] enhances the performance of naive CCN by adopting an MHT-based signing mechanism, which is a kind of aggregate signing method. In other words, CCNx divides a content by k segments which is the size of MHT, and applies MHT-based signing to a group of k segments. For example, CCNx first constructs a binary tree by taking the k segments as its leaf nodes, and then it signs the root node with typical digital signature primitive such as RSA-PSS. Since CCNx needs to buffer at least k data segments, they do not provide *on-the-fly data packet generation*. Jeff Burke et al. provided an HMAC-based data packet authentication mechanism in CCN to control the lighting in a building automation system [9,10]. Even though each data packet in their scheme can be verified independently, it requires system setup and the secure distribution of HMAC keys. Furthermore, HMAC-based authentication does not guarantee *non-deniability*. In addition, forwarding engines in CCN routers do not support HMAC-based packet authentication because HMAC-based methods require complex key management. TIM combines the concept of Trapdoor Hash Function (THF) and MHT for efficient signature signing and verification [13]. TIM can efficiently compute a signature of a data packet by using the feature of THF with a few field multiplications. In addition, data packets can be efficiently verified in the requester-side by using the property of implicit authentication in MHT. However, since TIM makes use of MHT, a content publisher needs to buffer at least k data segments for signing them. Thus, TIM does not provide *on-the-fly data packet generation* well. TLDA is a two-layered authentication mechanism for CCN and content in TLDA is encoded with a data part and meta part [14]. The data part conveys the real data segments and the meta part conveys hash values and a signature for authenticating data segments in the data part. Since TLDA requires an encoding process before sending content, it is more suitable for constant data such as installation and VoD files rather than dynamic data such as online streaming. Furthermore, the transmission of Meta part segments need to precede the data part segments. Therefore, TLDA lacks *on-the-fly data packet generation* and *verification independence*. Differently from the existing data packet authentication mechanisms, the proposed mechanism satisfies all requirements with the concept of overlaid signing.

In view of the verification cost, the proposed mechanism can significantly reduce the overhead for verifying the certificate compared with naive CCN, LIVE, CCNx, and TIM. This is why the CCN routers in our scheme do not depend on the external trusted authorities (TAs), which reduces the cost of verifying the certificate of a public key from a linearly increased cost $n_R C_{cert}$ to a constant cost C_{cert}. In the case of typical PKI-based methods, the request to verify a certificate needs to be routed to TAs. Thus, it depends on the condition of the routing path and the number of nodes in the path. Note that the proposed scheme requires the generation of a new signature at the CCN router which first receives the data packet. However, the cost of generating a signature is much lower than the cost of verifying a certificate.

Table 1. Features Comparison.

	Verification Independence	On-the-fly Generation	Non Deniability	Setup Freeness	Used Technique	Verification Cost for Forwarding Engines
Naive CCN	O	O	O	O	Per-Packet Signing	$n_R C_{cert}+n_R C_{ver}$
LIVE [11]	O	X	X	X	OTS	$n_R C_{cert}+n_R C_{ver}$
[9,10]	O	O	X	X	HMAC	Cannot Support
CCNx [1–3]	O	X	O	O	MHT	$n_R C_{cert}+n_R C_{sv}+n_R C_{ver}$
TIM [13]	O	X	O	O	THF, MHT	$n_R C_{cert}+n_R C_{sv}+n_R C_{thf}+n_R C_{ver}$
TLDA [14]	X	X	O	O	MHT	Cannot Support
This work	O	O	O	O	Overlaid Signing	$C_{cert}+C_{sig}+n_R C_{ver}$

OTS: One-time signature, THF: Trapdoor hash function, MHT: Merkle hash tree; n_R: average number of forwarding engines from a publisher to a requester, C_{cert}: cost of certificate; validation, C_s: cost of signing, C_v: cost of signature verification, C_{sv}: cost of verifying a sibling path; for a MHT, C_{thf}: cost of trapdoor hash function.

6.2. Performance Analysis

In this section, the computational overhead of the proposed method is analyzed and compared with the method currently used in CCNx, which is the reference implementation of CCN. Even though there exists simulation tool on NDN such as ndnSIM [17], we implement the proposed mechanism on the actual CCNx reference implementation. The prototype of the proposed method is implemented by using FIPS-OpenSSL v2.0.16 on a Laptop with Intel(R) Xeon CPU E3-1535M v5 running 2.90 GHz. It is assumed that Online Certificate Status Protocol (OCSP) is used to verify the validity of publisher's certificate, which is embedded in the content packets. According to the data from several OCSP providers [18], it is known that a certificate verification usually requires 100 ms (0.1 s) through the OCSP protocol (s and ms mean second and millisecond, respectively).

Table 2 compares the performance of the proposed method with the method currently used in CCN. For the comparison, we make use of three digital signature algorithms such as RSA-PSS, DSA, and ECDSA. When RSA-PSS with a 2048-bit key is used as an underlying digital signature algorithm, signing a packet and verifying it requires 0.004902 and 0.000146 s, respectively. Thus, RE_1, the first routing entity receiving content packets, consumes 0.105048 s to verify the certificate (0.1 s) and a received content packet (0.000146 s), and resigning it (0.004902 s). Other routing entities in the routing path consume 0.000146 s, since they only need to verify the received content packet. When DSA using a 2048-bit ($|p| = 2048$, $|q| = 224$) key is used as the digital signature algorithm, signing a packet and verifying it consume 0.00197 and 0.00242 s, respectively. Signing a packet and verifying it with ECDSA (P-224 curve) require 0.000353 and 0.000741 s, respectively. Thus, RE_1 with DSA 2048 (resp. ECDSA 224), the first routing entity receiving content packets, consumes 0.104390 s (resp. 0.101094 s) to verify the certificate (0.1 s) and a received content packet in 0.00242 s (resp. 0.000741 s), and resigning it in 0.00197 s (resp. 0.000353 s). Other RE_i ($i \neq 1$) routing entities in the routing path require 0.00242 s (resp. 0.000741 s) to verify each received packet.

The performances of the original CCN with the use of RSA-PSS 2048, DSA 2048, ECDSA 224 or RSA-PSS ($|n| = 2048$) are also measured. Since every routing entity in the routing path executes a PKI-based content verification, they require the cost of verifying the embedded certificate and the received content packet. DSA, ECDSA, and RSA-PSS each require 0.002420, 0.000741, and 0.000146 s for their signature verification. Thus, the original CCN using DSA, ECDSA, and RSA-PSS requires 0.102420, 0.100741, and 0.100146 s, respectively, to verify a content packet. Even though RE_1 requires a slightly larger overhead than the original CCN (about 4.9%, 1.9%, and 0.4% slower than the original CCN for

RSA-PSS, DSA, and ECDSA, respectively), the cost of content packer verification in other routing entities in the routing path is reduced by hundreds of factors. Our experimental results support the theoretical analysis described in Section 6.

Table 2. Performance Comparison of Data Packet Verification in Routing Entities (In the proposed method, the first routing entity(RE_1) receiving the content packet from the publisher conducts signature verification and signature generation, while other routing entities(RE_i for $i \neq 1$) require signature verification. The timing cost is the average of 100 executions). In experiments, RSA-PSS, DSA, and ECDSA use security parameter of $|n| = 2048$, P-224 curve, and ($|p| = 2048$, $|q| = 224$), respectively.

Methods	Time (S)	Operations
Naive CCN with RSA-PSS	0.100146 s	Verify *Certificate* and *RSA-PSS*
Naive CCN with DSA	0.102420 s	Verify *Certificate* and *DSA*
Naive CCN with ECDSA	0.100741 s	Verify *Certificate* and *ECDSA*
RE_1 with RSA-PSS *(This work)*	0.105048 s	Verify *Certificate* and Generate *RSA-PSS*
RE_1 with DSA *(This work)*	0.104390 s	Verify *Certificate* and Generate *RSA-PSS*
RE_1 with ECDSA *(This work)*	0.101094 s	Verify *Certificate* and Generate *RSA-PSS*
RE_i with RSA-PSS *(This work)*	0.000146 s	Verify *RSA-PSS*
RE_i with DSA *(This work)*	0.002420 s	Verify *RSA-PSS*
RE_i with ECDSA *(This work)*	0.000741 s	Verify *RSA-PSS*

When a user requests a data packet, several routing entities participate in the forwarding procedure. In Table 2, the number of routing entities is not considered when measuring the cost of computation. Therefore, we present a comparative analysis that takes into account the number of routing entities involved. In Figure A1 in Appendix A, the total cost for data packet authentication in CCN architecture is measured by taking the number of routing entities into account. The proposed mechanism requires a much lower total timing cost compared with the original CCN packet verification mechanism when RSA-PSS, DSA, and ECDSA are used for the underlying digital signature algorithm. Note that for up to four routing entities, the proposed mechanism using ECDSA 224 digital signature provides the best performance. However, as the number of routing entities increases, using RSA-PSS results in a better performance. This is because the verification cost of RSA-PSS is much lower than that of DSA and ECDSA. (Using ECDSA can reduce packet overhead because the key size and signature size of ECDSA is much lower than RSA-PSS.)

In Figure A2 in Appendix A, the average cost for data-packet-processing in each routing entity is also presented. In each case, it is assumed that publishers use the same signature scheme. For example, in Figure A2, "*Naive-CCN with DSA*" means the scenario where all publishers use DSA for data packet authentication. Since the cost of communication overhead cannot be accurately measured, we focus only on the time for verification. As seen in the figure, our architecture outperforms other approaches, and the difference increases according to the number of routing entities. From Figure A2, the average cost for data packet processing in each routing can be obtained. In Figure A2, to evaluate the average cost, the total cost is divided by the number of routing entities. Similar to the result of Figure A1, the average cost of the proposed mechanism is much lower than that of the original CCN mechanism and the performance improvement increases as the number of the related routing entities increase. With respect to average

timing cost, using RSA-PSS contributes the best performance because the signature verification of RSA-PSS is much faster than that of DSA and ECDSA.

7. Conclusions

In this paper, the necessity of data packet verification in forwarding engines is first identified, and a new authentication method supporting a faster data packet verification is proposed by designing a two-layered authentication architecture. The proposed method provides efficient data packet verification while maintaining the advantages of the original CCN compared to the existing technologies. Note that only the proposed method supports all useful features of the naive CCN and its performance is highly improved compared to the naive CCN. In existing techniques, it was not easy to perform data verification in all forwarding engines due to the cost of verification. In this work, a new mothod is proposed which dramatically reduces the cost of verification in forwarding engines, and so it can prevent many attacks that cannot be prevented when forwarding engines do not verify all data packets. As further research topics, we plan to extend the proposed technique to the mobile environment by considering mobile-specific threats [19] and to apply the AI-based approach to enhance performance and security [20].

Author Contributions: Conceptualization and methodology, T.-Y.Y.; validation, S.C.S.; formal analysis, T.-Y.Y.; investigation, J.K. and D.M.; writing—original draft preparation, T.-Y.Y. and S.C.S.; writing—review and editing, T.-Y.Y., J.K., D.M. and S.C.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the MHW (Ministry of Health and Welfare), Korea (Grant number: HI19C0842) supervised by the KHIDI (Korea Health Industry Development Institute) and the project title is DisTIL (Development of Privacy-Reinforcing Distributed Transfer-Iterative Learning Algorithm).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

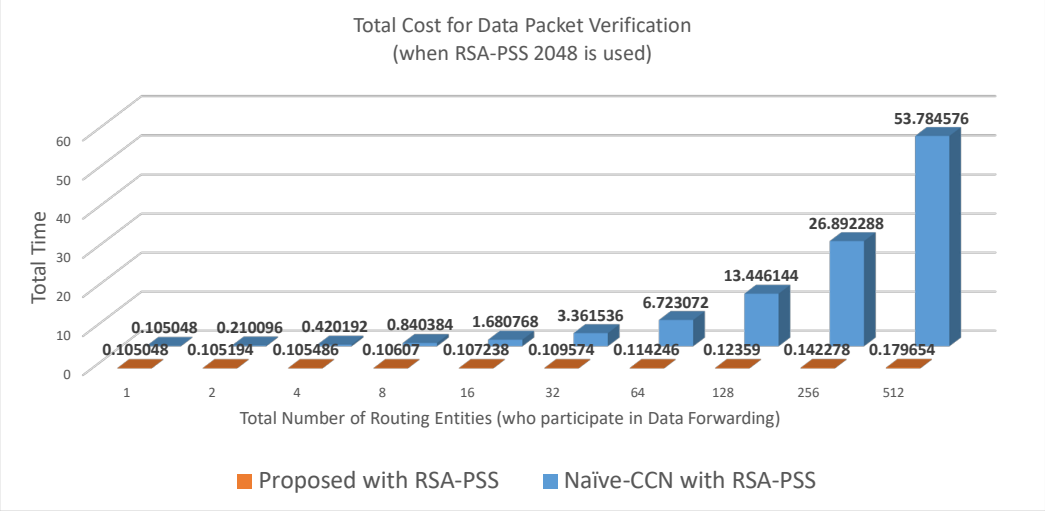

Figure A1. *Cont.*

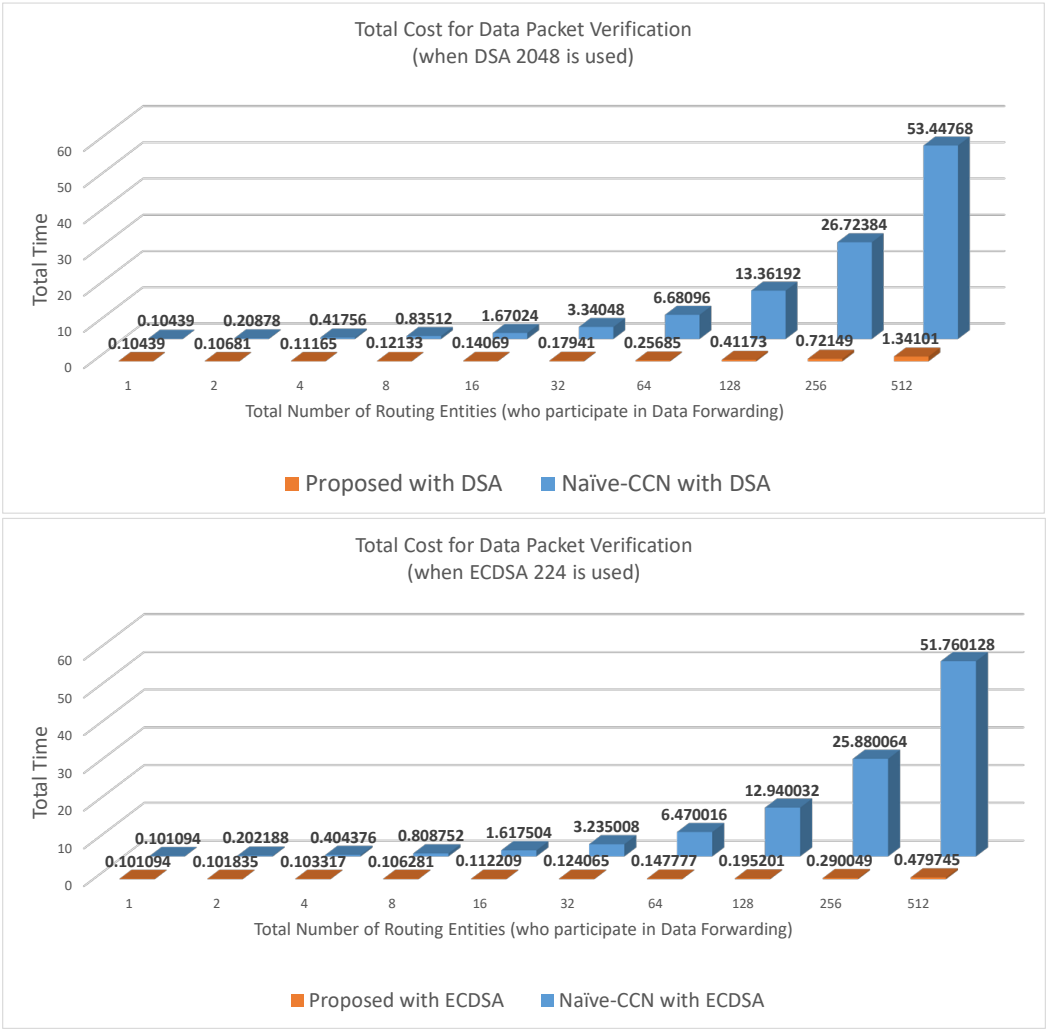

Figure A1. Total timing cost comparison for authenticating data packets.

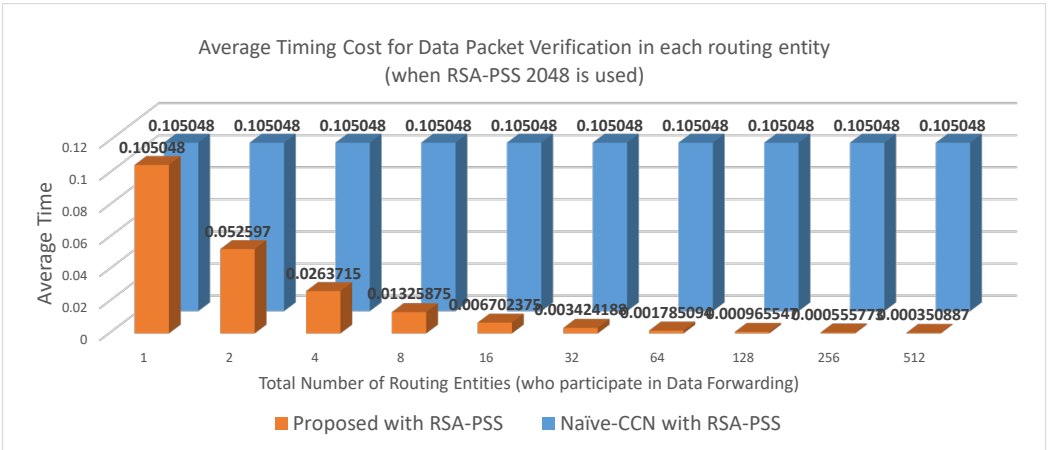

Figure A2. *Cont.*

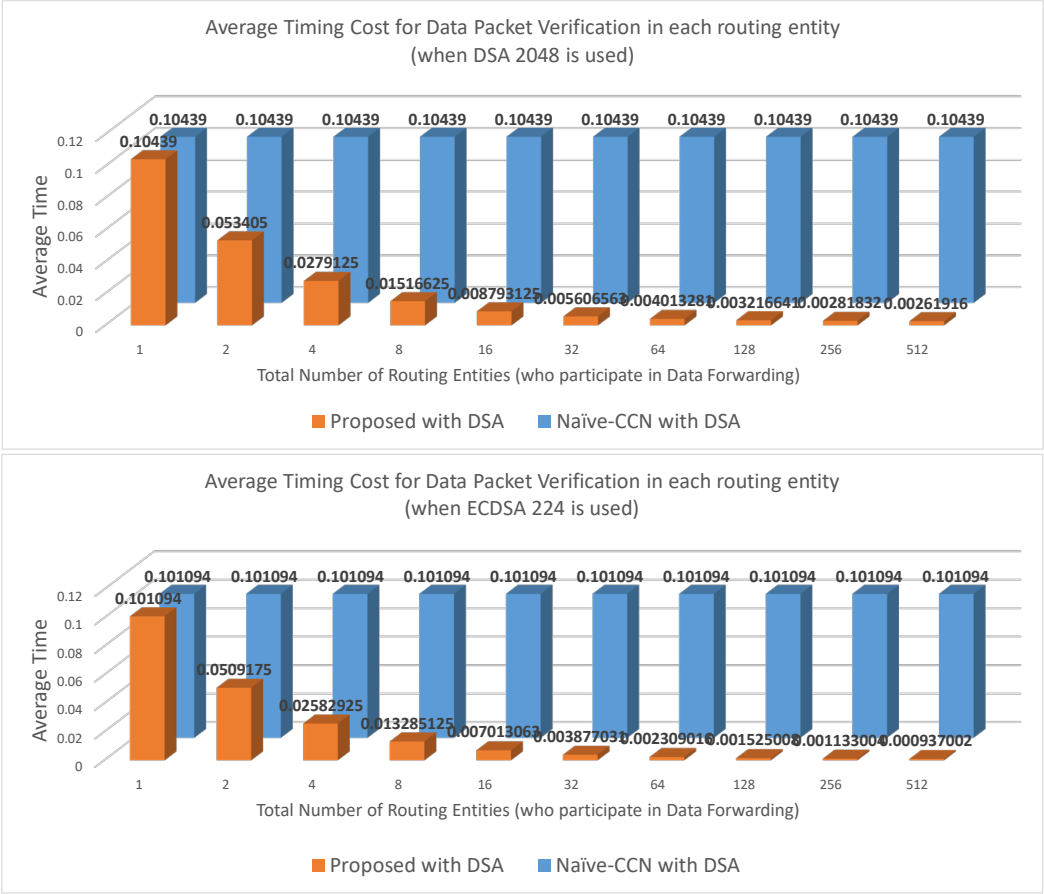

Figure A2. Average timing cost comparison in each router for authenticating data packets.

References

1. Jacobson, V.; Smetters, D.; Thornton, J.; Plass, M.; Briggs, N.; Braynard, R. Networking Named Content. In Proceedings of the 5th international conference on Emerging networking experiments and technologies(CoNEXT'09), Rome, Italy, 1–4 December 2009.
2. Smetters, D.; Jacobson, V. *Securing Network Content*; PARC Technical Report; PARC: Palo Alto, CA, USA, 2009.
3. Named Data Networking (NDN) Project. Available online: https://named-data.net (accessed on 20 October 2020).
4. CCNx Project. Available online: http://github.com/ProjectCCNx/ccnx (accessed on 20 October 2020).
5. Merkle, R. A Digital Signature Based on a Conventional Encryption Function. In Proceedings of the Advances in cryptology (CRYPTO'87), LNCS 293, Barbara, CA, USA, 11–15 August 1987; pp. 369–378.
6. Merkle, R. A Certified Digital Signature. In Proceedings of the Advances in cryptology (CRYPTO'89), LNCS 435, Barbara, CA, USA, 23–27 August 1989; pp. 218–238.
7. Baugher, M.; Davie, B.; Narayanan, A.; Oran, D. Self-Verifying Names for Read-Only Named Data. In Proceedings of the IEEE INFOCOM 2012 Workshop on Emerging Design Choices in Named-Oriented Networking, Newark, NJ, USA, 30 March 2012; pp. 274–279.
8. Moiseenko, I. Fetching Content in Named Data Networking with Embedded Manifests; NDN Technical Report NDN-0025; 2014. Available online: https://named-data.net/wp-content/uploads/2014/09/ndn-tr-25-manifest-embedding.pdf (accessed on 20 October 2020).

9. Burke, J.; Horn, A.; Marianantoni, A. Authenticated Lighting Control Using Named Data Networking; NDN Technical Report NDN-011 Rev.1; 2012. Available online: https://named-data.net/wp-content/uploads/TRlighting.pdf (accessed on 20 October 2020).
10. Burke, J.; Gasti, P.; Nathan, N.; Tsudik, G. 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, Turin, Italy, 14–19 April 2013; pp. 393–398.
11. Li, Q.; Zhang, X.; Zheng, Q.; Sandhu, R.; Fu, X. LIVE: Lightweight integrity verification and content access control for named data networking. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 308–320. [CrossRef]
12. Refaei, T.; Horvath, M.; Schumaker, M.; Hager, C. Data Authentication for NDN using Hash Chains. In Proceedings of the IEEE Symposium on Computers and Communication (ISCC), Messina, Italy, 27–30 June 2016; pp. 982–987.
13. Seo, S.C.; Youn, T. TIM: A Trapdoor Hash Function-based Authentication Mechanism for Streaming Authentication. *KSII Trans. Internet Inf. Syst.* **2018**, *12*, 2922–2945. [CrossRef]
14. Seo, S.C.; Youn, T. TLDA: An Efficient Two-Layered Data Authentication Mechanism for Content-Centric Networking. *Hindawi Secur. Commun. Netw.* **2018**, *2018*, 5429798. [CrossRef]
15. Zhang, Z.; Yu, Y.; Zhang, H.; Newberry, E.; Mastorakis, S.; Li, Y.; Afanasyev, A.; Zhang, L. An Overview of Security Support in Named Data Networking; Technical Report NDN-0057. Available online: http://named-data.net/techreports.html (accessed on 20 October 2020).
16. Gasti, P.; Tsudik, G.; Uzun, E.; Zhang, L. Dos and DDoS in Named Data Networking. In Proceedings of the 22nd International Conference on Computer Communications and Networks (ICCCN), Nassau, Bahamas, 30 July–2 August 2013; pp. 1–7.
17. Mastorakis, S.; Afanasyev, A.; Zhang, L. On the Evolution of ndnSIM: An Open-Source Simulator for NDN Experimentation. *ACM Comput. Commun. Rev.* **2017**, *47*, 19–33. [CrossRef]
18. OCSP Response Time. Available online: https://www.digicert.com/blog/ocsp-times-and-what-they-mean-for-you/ (accessed on 20 October 2020).
19. Qamar, A.; Karim, A.; Chang, V. Mobile malware attacks: Review, taxonomy & future directions. *Future Gener. Comput. Syst.* **2019**, *97*, 887–909.
20. Sun, J.; Zhang, Y.; Liao, D.; Sun, G.; Chang, V. AI-based survivable design for hybrid virtual networks for single regional failures in cloud data centers. *Clust. Comput.* **2019**, *22*, 12009–12019. [CrossRef]