Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 8071267, 12 pages
https://doi.org/10.1155/2018/8071267

Research Article

Secure and DoS-Resilient Fragment Authentication in CCN-Based Vehicular Networks

Sangwon Hyun and Hyoungshick Kim

Department of Software, Sungkyunkwan University, Suwon 16419, Republic of Korea

Correspondence should be addressed to Hyoungshick Kim; hyoung@skku.edu

Received 31 August 2017; Revised 31 December 2017; Accepted 3 January 2018; Published 6 February 2018

Academic Editor: Antonella Molinaro

Content-Centric Networking (CCN) is considered as a promising alternative to traditional IP-based networking for vehicle-to-everything communication environments. In general, CCN packets must be fragmented and reassembled based on the Maximum Transmission Unit (MTU) size of the content delivery path. It is thus challenging to securely protect fragmented packets against attackers who intentionally inject malicious fragments to disrupt normal services on CCN-based vehicular networks. This paper presents a new secure content fragmentation method that is resistant to Denial-of-Service (DoS) attacks in CCN-based vehicular networks. Our approach guarantees the authenticity of each fragment through the immediate fragment verification at interim nodes on the routing path. Our experiment results demonstrate that the proposed approach provides much stronger security than the existing approach named FIGOA, without imposing a significant overhead in the process. The proposed method achieves a high immediate verification probability of 98.2% on average, which is 52% higher than that of FIGOA, while requiring only 14% more fragments than FIGOA.

1. Introduction

Smart transportation based on vehicle-to-everything (V2X) communications is considered as a promising technology due to its potential to provide intelligent services for driving safety and efficiency. Due to the limitation of traditional IP-based networking in vehicular network environments, the transition to a new network paradigm is being discussed. As one of these research directions, several previous studies have proposed various possible Content-Centric Networking (CCN) architectures for V2X communication environments [1–3].

CCN [4] was originally proposed as a future Internet architecture that provides *named data*, *name-based request and routing*, and *in-network data caching*. Those characteristics seem to be well suited to V2X communications that require high mobility and intermittent connectivity of vehicles. In other words, CCN allows a vehicle to retrieve interested data from any nearby entities even though the vehicle has no connectivity with the original data publisher. Because V2X communication systems are generally designed for the vehicle and pedestrian communication to guarantee the safety of people, establishing secure communication between network entities has naturally become the most critical issue for V2X communications.

In this paper, we particularly focus on the problem to provide the authenticity of packet fragments in CCN. To apply the CCN architecture to V2X communication services, it is important to mitigate various network attacks. For example, an attacker can generate a massive amount of fake fragments [5] to disrupt normal services on V2X communication. A possible solution for checking the authenticity of incoming fragments is to use a digital signature system. CCN already mandates that every content be digitally signed by its publisher, and this allows a receiving node to verify the authenticity of the received content through the signature verification. However, each content packet in CCN can be fragmented according to the Maximum Transmission Unit (MTU) size of the data delivery path. Therefore, checking the authenticity of incoming fragments at the content level is not efficient because this process typically incurs a significant delay to collect necessary fragments and reassemble them.

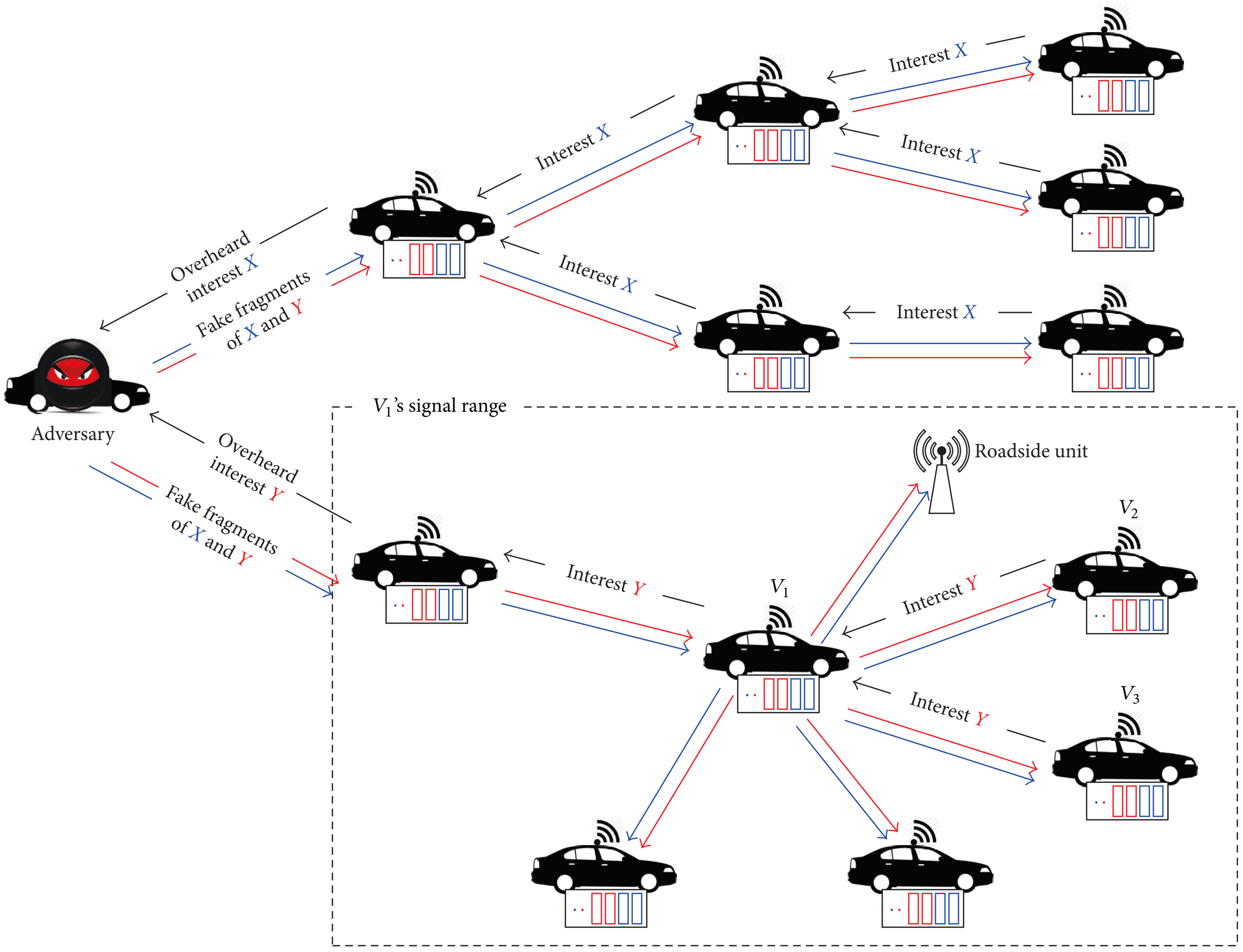


Figure 1: DoS attack against FIGOA in CCN-based vehicular network.

These delays are even accumulated at every hop on the routing path. A better option that avoids such latency is *fragment-based* authentication rather than content-based one, so that each fragment whose authenticity has been confirmed can efficiently be propagated to the next hop without waiting for the remaining fragments.

FIGOA [5] is a recent approach to fragment-based authentication in CCN that integrates a one-way hash chain and signature. However, due to the use of one-way hash chain, any single fragment loss results in that none of the subsequent fragments on the hash chain can be verified immediately upon receipt. Consequently, propagation delays can be significant with unverifiable fragments for some V2X applications requiring timely delivery of data packets (e.g., the message for a fatal road accident). To prevent such authentication delays from leading to propagation delays, FIGOA takes a strategy of just forwarding fragments whose authenticity has not been confirmed yet. Unfortunately, this opens the door to DoS attacks to exhaust the fragment buffers of network nodes (e.g., vehicles and roadside units) by intentionally introducing a massive amount of immediately unverifiable fake fragments. Furthermore, CCN-based vehicular networking typically requires network nodes to cache and forward overheard unsolicited data as well as solicited data to facilitate data discovery and delivery under the mobility of vehicles [6–10], and this feature even increases the impact of such DoS attacks. To make matters worse, such flooding of fragments also makes it infeasible to identify the origin of the attack.

Figure 1 illustrates the DoS attack against FIGOA in the CCN-based vehicular network. The adversary who overhears the interest messages requesting the contents (e.g., X and Y) injects a massive amount of immediately unverifiable fake fragments of the requested contents. According to the optimistic forwarding of FIGOA, the unverified fake fragments are propagated over the vehicles and used to consume the fragment buffers of the vehicles. As particularly illustrated with V_1 in Figure 1, the unverified fragments broadcast by V_1 affect all the remaining nodes within V_1's signal range as well as V_2 and V_3 that requested the contents.

In this paper, we present the design and analysis of an efficient, secure, and DoS-resilient fragment authentication technique for CCN-based vehicular networks. The proposed technique not only guarantees the authenticity of fragments but also provides resistance to DoS attacks exploiting authentication delays. Our key contribution is to propose a novel hash tree construction, which seamlessly integrates an erasure code (for forward error correction) into a hash tree. The proposed hash tree achieves a high immediate verification probability for each fragment upon receipt. Specifically, our analysis results demonstrate that our approach achieves a high immediate verification probability of 98.2% on average over all the test cases, which is 52% higher than that of FIGOA, while requiring 14% more fragments than FIGOA.

As another layer of protection against DoS attacks, we have devised an adaptive forwarding strategy for fragments whose authenticity cannot be confirmed immediately upon receipt. By adaptively adjusting the probability of forwarding such fragments according to the recently observed ratio of authentic/inauthentic fragments, it allows the rapid propagation of legal fragments in benign environments, while effectively filtering out illegal fragments in hostile environments.

The rest of this paper is organized as follows. Section 2 provides some preliminary information on CCN, hash tree, and Rabin's Information Dispersal Algorithm. Section 3 presents the proposed techniques for secure and DoS-resilient fragment authentication and delivery. Section 4 describes the analysis results of the security and efficiency of the proposed approach. Section 5 overviews related work, and Section 6 concludes this paper.

2. Background

2.1. Overview of CCN. Every CCN packet contains a content name instead of a host identifier. In particular, a user requests a content by issuing an *interest* that contains the name of the content. Each CCN node performs content name-based routing for this interest, so that it can be delivered to the corresponding content publisher. Upon receiving the interest, the publisher replies back with the requested content, and the content is delivered to the user by following the reverse path of the interest via the CCN's routing mechanism. During delivery, the content is fragmented according to the MTU size of the network link. In addition, each CCN node performs in-network caching to reduce redundant data transmissions and latency. To enable the user to confirm the authenticity of a received content, every content is appended with the publisher's digital signature.

2.2. Hash Tree. We adopt the general scheme of a hash tree described in [11]. A hash tree is constructed over multiple pieces of data to be authenticated, where a leaf node in the tree holds each data piece. Each internal node stores a set of hash values, each of which is the result of applying a collision-resistant hash function (e.g., SHA-256) to the value(s) stored at each of its child nodes. The hash value at the root node is signed using a digital signature algorithm such as RSA, DSA, or ECDSA. Figure 2 presents an example of a hash tree to authenticate eight data pieces, $V_{0,0}$~$V_{0,7}$. We assume that each

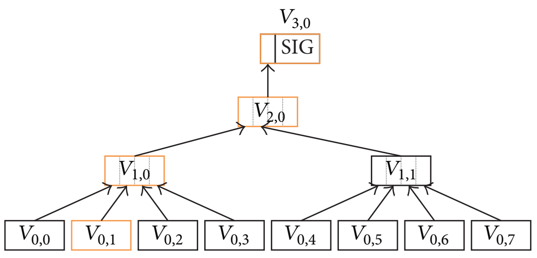


FIGURE 2: Hash tree to authenticate eight pieces of data denoted as $V_{0,0}$~$V_{0,7}$.

internal node in this tree can have up to four hash values or a single digital signature along with a hash value. We refer to the path from each node up to the root node as its *authentication path*. In Figure 2, for example, the authentication path of $V_{0,1}$ consists of $V_{0,1}$, $V_{1,0}$, $V_{2,0}$, and $V_{3,0}$. Given all the nodes over an authentication path, it is possible to verify their authenticity by performing signature verification on the root node and repeatedly computing and comparing hash values along the authentication path. When a piece of data corresponding to a leaf node (i.e., $V_{0,0}$~$V_{0,7}$) is modified, the hash values stored at its ancestor nodes, including the root node, are all changed. Moreover, due to the collision-resistance of the hash function, it is not feasible to modify any piece of data without the sign key. Thus, all of the data pieces are authenticated with just a single signature (SIG).

2.3. Information Dispersal Algorithm. We use *Rabin's Information Dispersal Algorithm* (IDA) [12] to enable the loss-resilient recovery of the hash values stored at internal nodes in a hash tree. Rabin's IDA is a type of erasure code and consists of two operations: *Dispersal* and *Recovery*. For a given *Data* chunk consisting of m equal-length blocks, $Dispersal(Data, m, n)$ generates n encoded blocks EB_i $(1 \leq i \leq n,\ n \geq m)$, each of which has the same length as an input block. Given any m out of n encoded blocks, $Recovery(\{\mathrm{EB}_{i_j} \mid 1 \leq i_j \leq n,\ 1 \leq j \leq m\}, m, n)$ reconstructs the original *Data*. Thus, up to $(n - m)$ block losses can be tolerated by using IDA. We refer to the ratio $(n - m)/n$ as the *redundancy rate* of IDA.

3. Our Approach

This section presents the proposed techniques for achieving secure and DoS-resilient fragment authentication in CCN. The key contribution is to apply the systematic IDA into a hash tree so that the system becomes more resilient to time delays and packet losses. In the existing hash tree structure described in Section 2.2, missing any internal node disables immediate verification of its child nodes upon receipt due to the loss of the hash values of the child nodes. To address this problem, we apply IDA for the hash values at each height of the hash tree, and this enables immediate verification of child nodes upon receipt even under a certain degree of losses by recovering missing hash values in a prompt manner. Moreover, the systematic property allows the hash values included in the internal nodes to directly appear in the encoded results generated by the dispersal operation. Thus if some of the hash values are received correctly, then we do not need to perform the recovery operation for the received hash values. This technique can significantly reduce the computation cost at receivers. In addition, receiving some of the hash values at height i allows us to immediately verify the corresponding child nodes at height $i-1$ upon receipt without waiting for m_i nodes at height i to be received. This increases the chance of immediate verification in out-of-order delivery situations.

Notations summarizes the notations that will be used in the remainder of this paper.

3.1. Content Fragmentation and Authentication. Given a content object (CO) to be transmitted, we first divide it into equal-sized fragments. For the fragment size f, we typically use the MTU size of the routing path via which CO will be delivered or if available the globally smallest MTU. Next, we construct a d-nary hash tree over the fragments of CO as described in Section 2.2. Each node in the hash tree corresponds to one fragment, and so we will use the terms "node" and "fragment" interchangeably hereafter. The hash tree illustrated in Figure 2 represents the case that CO consists of eight fragments, denoted as $V_{0,0} \sim V_{0,7}$, and each internal node can have up to four children's hash values ($d = \lfloor f/|F(\cdot)| \rfloor = 4$).

Unfortunately, in such a hash tree structure, any missing internal fragment results in the failure to construct the authentication path of all its child fragments. Thus, it becomes infeasible to immediately verify those child fragments upon receipt. To solve this problem, we integrate Rabin's IDA into the hash tree construction to add some extra redundant fragments to the internal fragments at each height of the tree, ensuring that we can tolerate up to a certain degree of fragment losses.

Systematic IDA. Directly applying Rabin's IDA is not efficient due to the mandatory requirement of the recovery procedure. In other words, a receiver must always perform the recovery procedure in order to recover the original data from the received m or more packets. We propose customizing IDA such that the original data directly appears on the encoded blocks. As a result, it is possible to skip the recovery procedure as long as all of the original data blocks are given correctly. In other words, a receiver performs the recovery procedure only when some original data blocks are lost. Such a coding scheme that includes the original input in the encoded output is called *systematic code*. Thus, we refer to the two procedures of our customized IDA as *SystematicDispersal* and *SystematicRecovery*, respectively, in the remainder of this paper.

To achieve the above property, we adopt *Vandermonde matrix* [13] as the encoding matrix of IDA. Let $\mathbf{V} = [v_{ij}]_{n\times m}$ be a $n\times m$ Vandermonde matrix ($n \geq m$), where $v_{ij} = x_i^{j-1}$ and all x_i's are nonzero elements of a finite field such as $GF(2^8)$. $\mathbf{V}$ has the following properties [13]:

(i) Any m rows out of n rows in $\mathbf{V}$ are linearly independent.

(ii) Even if any m rows in $\mathbf{V}$ are substituted with the rows of an $m \times m$ identity matrix, the modified matrix still has the above property.

Let $Data = b_1, b_2, \ldots, b_N$ be a chunk of N bytes, where b_i represents the ith byte in $Data$ ($0 \leq b_i \leq 255$). For simplicity, we assume that N is a multiple of m, and all computations in our customized IDA are performed in $GF(2^8)$.

To customize IDA into a systematic code, we have to properly arrange the input data, so that the input data directly appears as the first m encoded blocks. This leads to the customized encoding procedure $SystematicDispersal(Data, m, n)$:

(1) The input $Data$ is grouped into blocks of N/m bytes: $Data = (b_1, \ldots, b_{N/m}), (b_{N/m+1}, \ldots, b_{N/m+N/m}), \ldots, (b_{(m-1)(N/m)+1}, \ldots, b_{(m-1)(N/m)+N/m})$, and then it is arranged in an $m \times N/m$ matrix as follows:

$$\mathbf{M} = \begin{pmatrix} b_1 & \cdots & b_{N/m} \\ b_{N/m+1} & \cdots & b_{N/m+N/m} \\ \vdots & & \vdots \\ b_{(m-1)(N/m)+1} & \cdots & b_{(m-1)(N/m)+N/m} \end{pmatrix}. \quad (1)$$

Each row of $\mathbf{M}$ corresponds to each block of N/m bytes in $Data$.

(2) Select the following $n \times m$ Vandermonde matrix for encoding:

$$\mathbf{V} = \begin{pmatrix} \mathbf{a_1} \\ \mathbf{a_2} \\ \vdots \\ \mathbf{a_m} \\ \mathbf{a_{m+1}} \\ \vdots \\ \mathbf{a_n} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 \\ 1 & x_{m+1} & x_{m+1}^2 & \cdots & x_{m+1}^{m-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{m-1} \end{pmatrix}, \quad (2)$$

where x_i's ($(m+1) \leq i \leq n$) are different nonzero elements of $GF(2^8)$.

(3) Encode *Data* into n blocks (rows) EB_i $(1 \le i \le n)$ as follows:

$$\mathbf{V} \cdot \mathbf{M} = \begin{pmatrix} b_1 & \cdots & b_{N/m} \\ b_{N/m+1} & \cdots & b_{N/m+N/m} \\ \vdots & & \vdots \\ b_{(m-1)(N/m)+1} & \cdots & b_{(m-1)(N/m)+N/m} \\ p\left(x_{m+1}, 1\right) & \cdots & p\left(x_{m+1}, \dfrac{N}{m}\right) \\ \vdots & & \vdots \\ p\left(x_n, 1\right) & \cdots & p\left(x_n, \dfrac{N}{m}\right) \end{pmatrix} = \begin{pmatrix} \mathrm{EB}_1 \\ \mathrm{EB}_2 \\ \vdots \\ \mathrm{EB}_m \\ \mathrm{EB}_{m+1} \\ \vdots \\ \mathrm{EB}_n \end{pmatrix}, \quad (3)$$

where $p(x_i, j) = \sum_{k=0}^{m-1} x_i^k b_{j+k(N/m)}$. Each row of the resulting matrix corresponds to one encoded block. Therefore, the original *Data* is directly included in the first m encoded blocks. We refer to the first m blocks and the remaining $(n-m)$ blocks as *original* and *redundant* blocks, respectively.

Given any m encoded blocks EB_{i_j} $(1 \le i_j \le n,\ 1 \le j \le m)$, we can perform *SystematicRecovery*$(\{\mathrm{EB}_{i_j} \mid 1 \le i_j \le n,\ 1 \le j \le m\}, m, n)$ to reconstruct any missing original data block(s):

(1) Let $\mathbf{A}$ be the $m \times m$ matrix composed of the m row vectors in $\{\mathbf{a_1}, \dots, \mathbf{a_n}\}$ which are used to generate the given encoded blocks (step 2 in *SystematicDispersal*). $\mathbf{A}$ is invertible because of the linear independence condition on $\mathbf{a_i}$ $(1 \le i \le n)$. Compute $\mathbf{A}^{-1}$.

(2) Recover the original data using the given m encoded blocks and $\mathbf{A}^{-1}$ as follows:

$$\mathbf{M} = \mathbf{A}^{-1} \begin{pmatrix} \mathrm{EB}_{i_1} \\ \mathrm{EB}_{i_2} \\ \vdots \\ \mathrm{EB}_{i_m} \end{pmatrix}. \quad (4)$$

The systematic IDA allows us to tolerate up to $\min(m, n-m)$ original data block losses, and at the same time if all of the first m blocks are received correctly, the recovery procedure is no longer required. Moreover, if a network node receives some original data blocks correctly, it does not need to compute the corresponding rows in M (in step 2 of *SystematicRecovery*). This significantly reduces the computational cost at each receiver.

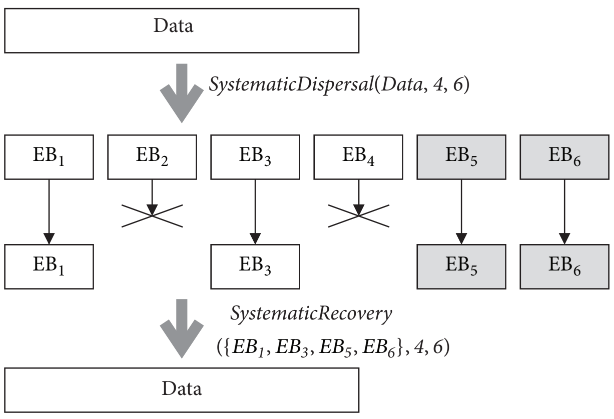


Figure 3: Example application of the systematic IDA.

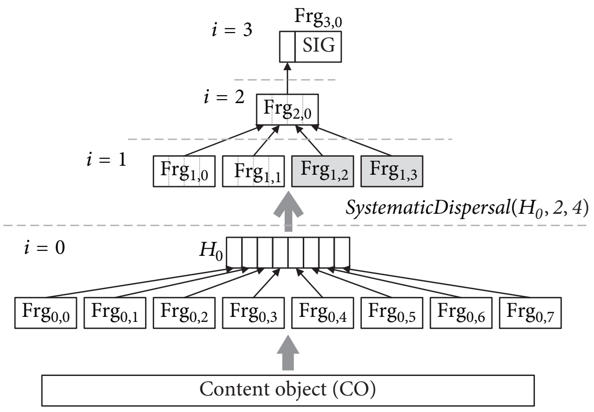


Figure 4: Proposed hash tree to authenticate eight content fragments.

Figure 3 illustrates an example application of the systematic IDA to the given data that is composed of four data blocks. Calling *SystematicDispersal*$(Data, 4, 6)$ generates the total of six encoded blocks: four original blocks EB_1~EB_4 and two redundant blocks EB_5 and EB_6. Even though two original blocks EB_2 and EB_4 are lost in transmission, they can be recovered from the four received encoded blocks by calling *SystematicRecovery*$(\{\mathrm{EB}_1, \mathrm{EB}_3, \mathrm{EB}_5, \mathrm{EB}_6\}, 4, 6)$.

Construction of Content Fragments. Now, we present the process to construct the proposed hash tree for a given CO, which is described in Algorithm 1. Figure 4 shows an example of the proposed hash tree constructed over CO which is composed of eight fragments. When IDA is not applied at height i (in lines (8)–(10) in Algorithm 1), $m_i = n_i$ (e.g., $m_0 = n_0 = 8$ in Figure 4), and n_i fragments are constructed by splitting *Data*. In line (15), we compute the hash values of each fragment, respectively, at height i, and concatenate them together to produce H_i

```
(1)  $Data \leftarrow \mathrm{CO}, i \leftarrow 0, end \leftarrow false$
(2)  while $end = false$ do
(3)      $m_i = \left\lceil \frac{|Data|}{f} \right\rceil$
(4)      $Data \leftarrow Data \parallel padding$                    $\triangleright |Data| = m_i f$
(5)      if $i > 0$ and $m_i > 1$ then
(6)          $n_i \leftarrow \min(\lceil m_i/(1-r) \rceil, d(m_i - 1))$ $\triangleright$ $r$: redundancy rate of IDA
(7)          Encode $Data$ into $n_i$ fragments $\mathrm{Frg}_{i,j}$ $(0 \le j < n_i)$ by
             performing $SystematicDispersal(Data, m_i, n_i)$.
(8)      else
(9)          $n_i = m_i$
(10)         Split $Data$ into $n_i$ fragments $\mathrm{Frg}_{i,j}$ $(0 \le j < n_i)$.
(11)     if $n_{i-1} = 1$ then
(12)         $\mathrm{Frg}_{i,0} \leftarrow Data \parallel Sign(Data) \parallel padding$
(13)         $end \leftarrow true$
(14)     else
(15)         $H_i \leftarrow F(\mathrm{Frg}_{i,0}) \parallel \cdots \parallel F(\mathrm{Frg}_{i,n_i-1})$
(16)         $Data \leftarrow H_i$
(17)         $i \leftarrow i + 1$
```

Algorithm 1: Proposed hash tree construction.

(i.e., $H_i = F(\mathrm{Frg}_{i,0}) \parallel \cdots \parallel F(\mathrm{Frg}_{i,n_i-1})$). For simplicity, we assume that $|H_i|$ is a multiple of f. Then H_i directly becomes $\mathrm{Frg}_{i+1,0}$~$\mathrm{Frg}_{i+1,m_{i+1}-1}$ at the upper height $i+1$, where $m_{i+1} = \lceil |H_i|/f \rceil$. Next, we generate additional redundant fragments $\mathrm{Frg}_{i+1,m_{i+1}}$~$\mathrm{Frg}_{i+1,n_{i+1}-1}$ at height $i+1$ by calling $SystematicDispersal(H_i, m_{i+1}, n_{i+1})$ in line (7). In Figure 4, for example, $H_0 = F(\mathrm{Frg}_{0,0}) \parallel \cdots \parallel F(\mathrm{Frg}_{0,7}) = \mathrm{Frg}_{1,0} \parallel \mathrm{Frg}_{1,1}$, and calling $SystematicDispersal(H_0, 2\,(= m_1), 4\,(= n_1))$ additionally generates $\mathrm{Frg}_{1,2}$ and $\mathrm{Frg}_{1,3}$ at height 1. The recovery of any missing original fragment (e.g., either $\mathrm{Frg}_{1,0}$ or $\mathrm{Frg}_{1,1}$) is possible with any two fragments among $\mathrm{Frg}_{1,0}$~$\mathrm{Frg}_{1,3}$. That is, we can tolerate up to two fragment losses. We repeat the same process for each height of the tree until the tree converges into a single node/fragment at a certain height (e.g., $\mathrm{Frg}_{2,0}$ at height 2). We compute the hash value of that single fragment and its signature, and these finally form the root (e.g., $\mathrm{Frg}_{3,0}$) of the tree (in line (12)).

3.2. Transmitting and Verifying Fragments. Immediate verification of received fragments is an important capability for mitigating DoS attacks. To achieve this, ancestors in the hash tree must be received earlier than their children. One possible way to achieve this goal is to enable the sender to transmit the fragments at each height only after it confirms that the receiver has received a sufficient number ($\ge m_i$) of fragments at every upper height to recover the hash values there. In other words, the sender should wait for the acknowledgment of height i from the receiver before transmitting the fragments at the lower height ($i-1$). However, this approach may introduce some overheads (e.g., latency and message transmission) caused by the introduction of height-by-height acknowledgments.

In CCN, every fragment of the same content is labeled with the same content name, and this increases the chance that the fragments of the same content are delivered through the same routing path [14]. In consequence, these fragments are likely to arrive in the same order as transmission. To leverage this property, we transmit the fragments at height i earlier than those at height $i-1$, and for the same height i we transmit $\mathrm{Frg}_{i,0}$ to Frg_{i,n_i-1} in a sequential order. This transmission method increases the chance of immediate verification of fragments upon receipt. In addition, the use of IDA in our approach further increases the probability of immediate verification, even with the occurrence of fragment losses. Moreover, taking advantage of the systematic property of our customized IDA, as soon as receiving an original fragment with its hash value, it is possible to immediately verify any child fragments of the received fragment without performing the recovery procedure. This is also of benefit to increase the chance of immediate verification in out-of-order delivery situations.

The root and its only child fragments (e.g., $\mathrm{Frg}_{3,0}$ and $\mathrm{Frg}_{2,0}$) in the hash tree are critical points of failure, because missing any of these disables the immediate verification of all the remaining fragments. To make these resistant against fragment loss, we transmit $\lceil 1/(1-r) \rceil - 1$ extra redundant copies for each of them. This can tolerate up to a fragment loss rate of r.

Now, we describe the procedures to verify and forward received fragments. Notations summarizes the descriptions of $\mathbb{V}$, $\mathbb{V}_i$, and $\mathbb{W}$. For each incoming fragment $\mathrm{Frg}_{i,j}$, we first check if it contains a signature. If so, we verify the fragment through signature verification and append it to $\mathbb{V}$ only if it is valid. If $\mathrm{Frg}_{i,j}$ contains no signature, then we check if its parent fragment $\mathrm{Frg}_{i+1,\lfloor j/d \rfloor}$ is already in $\mathbb{V}$. In that case, we compute the hash value of $\mathrm{Frg}_{i,j}$ and compare it with the one stored at the parent fragment; $F'(\mathrm{Frg}_{i,j}) \overset{?}{=} (F(\mathrm{Frg}_{i,j})$ in $\mathrm{Frg}_{i+1,\lfloor j/d \rfloor})$. If both hash values are the same, then we append $\mathrm{Frg}_{i,j}$ to $\mathbb{V}$. After adding each fragment to $\mathbb{V}$, we also forward it to the next hop, while dropping all inauthentic fragments without forwarding and buffering. If the parent fragment of $\mathrm{Frg}_{i,j}$ is not in $\mathbb{V}$, $\mathrm{Frg}_{i,j}$ cannot be verified right away. We refer to this kind of fragments as *unverifiable* fragments and will discuss how to handle unverifiable fragments later in this section.

If $|\mathbb{V}_i| \geq m_i$ and some of $\text{Frg}_{i,0}$~Frg_{i,m_i-1} are missing, then we perform $\textit{SystematicRecovery}(\{\text{Frg}_{i,j_k} \mid 0 \leq j_k < n_i,\ 0 \leq k < m_i\}, m_i, n_i)$ to recover those that are missing. After these are recovered, we also append them to $\mathbb{V}$. In addition, if some of Frg_{i,m_i}~Frg_{i,n_i-1} are missing, then we also regenerate the missing redundant fragments for further forwarding by performing $\textit{SystematicDispersal}(\text{Frg}_{i,0} \parallel \cdots \parallel \text{Frg}_{i,m_i-1}, m_i, n_i)$.

In Figure 4, for example, let us assume that $\text{Frg}_{1,0}$ has arrived and $\text{Frg}_{2,0} \in \mathbb{V}$. In that case, it is possible to immediately verify $\text{Frg}_{1,0}$ by computing its hash value $F'(\text{Frg}_{1,0})$ and comparing it with $F(\text{Frg}_{1,0})$ in $\text{Frg}_{2,0}$. In addition, if only $\text{Frg}_{1,0}$ and $\text{Frg}_{1,2}$ are in $\mathbb{V}_1$, then we can recover $\text{Frg}_{1,1}$ by calling $\textit{SystematicRecovery}(\{\text{Frg}_{1,0}, \text{Frg}_{1,2}\}, 2, 4)$. Now, we are ready to immediately verify any of $\text{Frg}_{0,0}$~$\text{Frg}_{0,7}$ upon receipt.

Even though the proposed method increases the chance for immediate verification of fragments, unverifiable fragments can still occur in the proposed method. That is, incoming fragments can be categorized into the three groups (*authentic*, *inauthentic*, or *unverifiable*). To reduce the delay in processing unverifiable fragments, we proposed a strategy to make an adaptive decision on whether or not to buffer and/or further forward received unverifiable fragments based on the statistics of the authenticity of other recently received fragments.

For this strategy to work in practice, we count the numbers (x, y, and z defined in Notations) of authentic, inauthentic, and unverifiable fragments, respectively, received within a specified time interval. With the probability of $x/(x+y+z)$, we forward the received unverifiable fragment to the next hop and then buffer it to $\mathbb{W}$. In contrast, with the probability of $y/(x+y+z)$, we discard it without forwarding and buffering. Finally, with the probability of $z/(x+y+z)$, we only buffer it to $\mathbb{W}$ without forwarding. The principle behind this strategy is that, in benign environments, where almost every fragment is authentic, incoming unverifiable fragments are also highly likely to be authentic. Thus, it is reasonable to forward them rather than dropping them to avoid undesirable retransmissions and propagation delays. On the other hand, in hostile environments, unverifiable fragments are likely to be malicious, and so they should be filtered out with a high chance to prevent malicious fragments from being propagated over the network.

In order for the fragments in $\mathbb{W}$ to eventually be verified, whenever a new fragment is added to $\mathbb{V}$, we check if any of its child fragments is in $\mathbb{W}$. If this is the case, we try to verify their authenticity recursively along the authentication path.

3.3. Adjusting Fragment Size. In CCN, the fragments of some content stored in the intermediate node's cache could be possibly delivered through a routing path whose MTU size does not match the fragment size. Let f_{orig} and f_{new} denote the original fragment size and the new fragment size for the new routing path, respectively. In the case that $f_{\text{new}} > f_{\text{orig}}$, if $f_{\text{new}} \approx cf_{\text{orig}}$ ($c \geq 2$), we could construct a larger fragment by merging adjacent c fragments at the same height of the hash tree. To support the case that $f_{\text{new}} < f_{\text{orig}}$, the one-way hash chain-based approach of FIGOA [5] can be integrated into our hash tree as follows: each fragment in our hash tree is further split into multiple smaller fragments, and a one-way hash chain is constructed over the smaller fragments as in FIGOA.

4. Analysis

In this section, we analyze the security and efficiency of the proposed approach and compare the results with FIGOA. We use the same notations as in Notations.

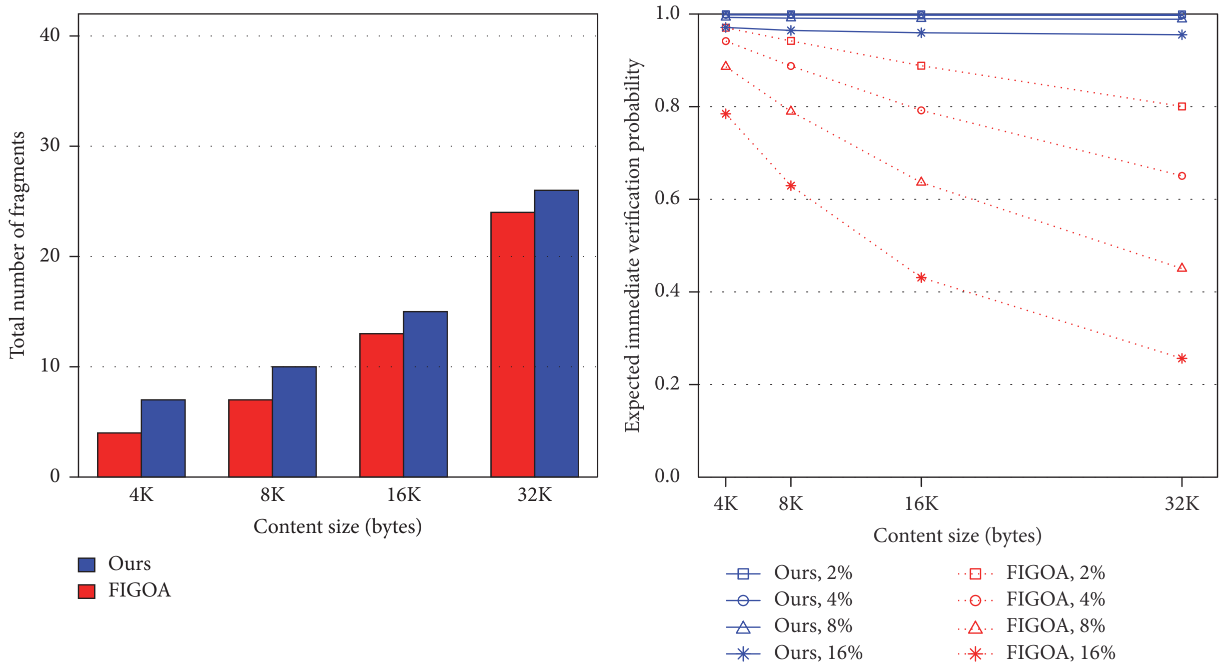
Communication Overhead. To investigate the communication overhead introduced by using the systematic IDA and the hash tree, we analyze the total number of fragments generated from the proposed hash tree for a given content.

Theorem 1. *Given a content object (CO), the total number of fragments generated from the proposed hash tree is* $N = \sum_{i=0}^{h} n_i = \lceil |\text{CO}|/f \rceil + \sum_{i=1}^{h-2} \min(\lceil m_i/(1-r) \rceil, d(m_i-1)) + \lceil 2/(1-r) \rceil$ *(see Notations for the meanings of h, d, m_i, n_i, r, and f).*

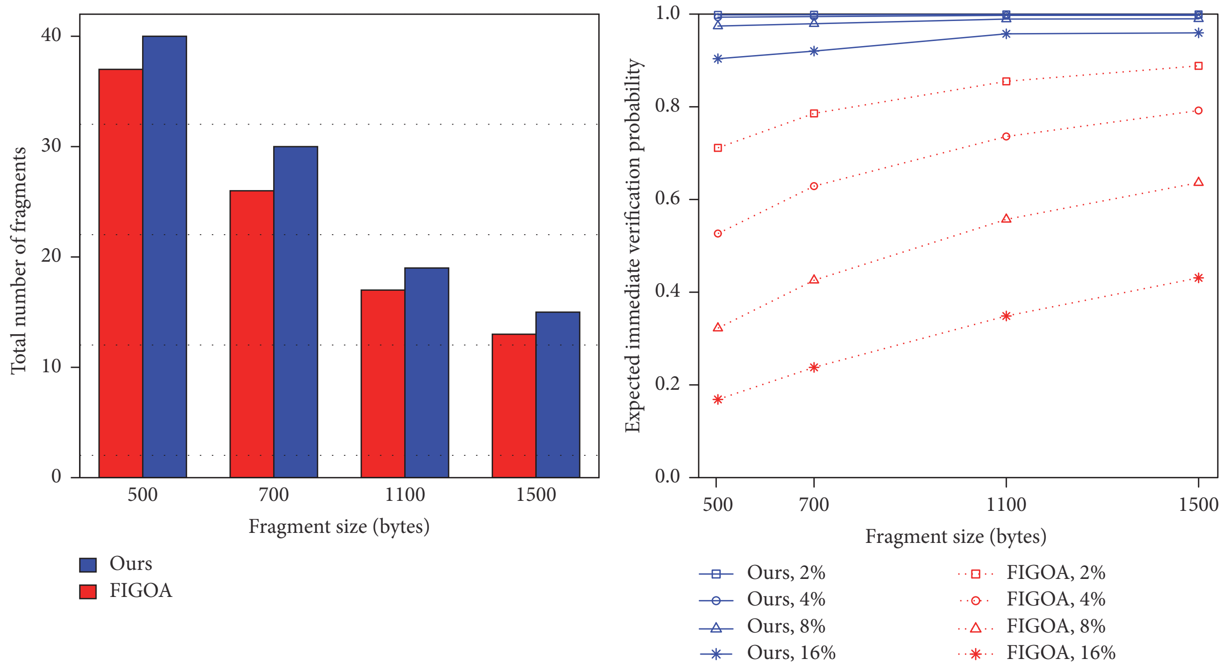
Proof. For height 0, $m_0 = n_0 = \lceil |\text{CO}|/f \rceil$. For each height i from 1 to $(h-2)$, $m_i = \lceil n_{i-1}/d \rceil$ and $n_i = \min(\lceil m_i/(1-r) \rceil, d(m_i-1))$; the term $d(m_i-1)$ is required to ensure that the hash tree eventually converges to the single root node. For both heights $(h-1)$ and h, $m_{h-1} = m_h = 1$ and $n_{h-1} = n_h = \lceil 1/(1-r) \rceil$. In consequence, the total number of fragments generated from our hash tree is $N = \sum_{i=0}^{h} n_i = \lceil |\text{CO}|/f \rceil + \sum_{i=1}^{h-2} \min(\lceil m_i/(1-r) \rceil, d(m_i-1)) + \lceil 2/(1-r) \rceil$. □

Computation Overhead. We analyze the computation cost required to verify a content object. For the entire hash tree constructed over the content object, a single signature verification is required to verify the root fragment. For all the remaining fragments, a single hash operation is required to verify each of them. Thus, at most, $\sum_{i=0}^{h-1} n_i$ hash operations are required in total.

Consider the computation cost to recover missing fragments. Recovery operations may be required for each height i from $h-2$ to 1 of the hash tree. Let l_i denote the number of missing original fragments at height i. Recovering l_i fragments first requires a single matrix inversion operation that costs $O(m_i l_i^2)$ [13]. Next, the cost to reconstruct l_i fragments is $O(m_i l_i)$, and this cost typically dominates the cost of matrix inversion because the single matrix inversion cost is amortized over l_i fragments [13]. l_i depends on network condition, and when the packet loss rate is p, $l_i = m_i p$ in the average case, and $l_i = \min(m_i, n_i - m_i)$ in the worst case. As a result, the cost of recovery operations is determined by m_i. In our experiments shown in Figure 5, for example, $2 \leq m_i \leq 5$



(a) Varying content sizes with the fixed fragment size of 1500 bytes



(b) Varying fragment sizes with the fixed content size of 16K bytes

FIGURE 5: Comparison between our approach and FIGOA by varying content size, fragment size, and fragment loss rate.

over all the test cases, and the cost of recovery operations is low for such small values of m_i.

Immediate Verification Probability. Immediate verification of received fragments can mitigate DoS attacks. Thus, in the following, we analyze the probability that each fragment in the proposed hash tree can be verified immediately upon receipt. Let $(1-p)$ denote the probability that any fragment is received correctly without any error or loss.

Theorem 2. *Upon receiving a fragment $Frg_{i,j}$ at height i ($0 \le i \le h$), the probability $f(i)$ that $Frg_{i,j}$ is immediately verifiable is $\prod_{u=i+1}^{h}\left\{1 - p\sum_{v=0}^{m_u-1}\binom{n_u-1}{v}(1-p)^{v}p^{n_u-1-v}\right\}$ for $0 \le i < h$, and $f(h) = 1$ (see Notations for the meanings of h, m_i, n_i, and p).*

Proof. In order for $\mathrm{Frg}_{i,j}$ to be immediately verifiable, at least one of the following two conditions should be satisfied for every height u ($u \ge i+1$) on the authentication path from

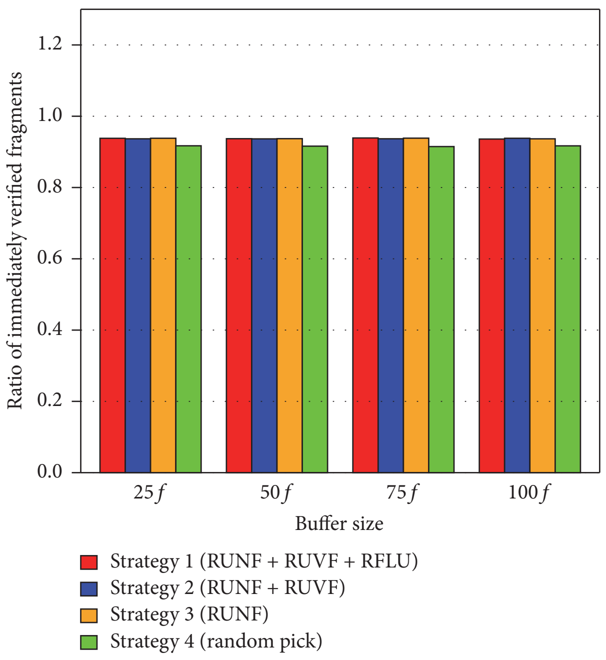


Figure 6: Simulation results of three different buffer replacement strategies (f: fragment size).

the node corresponding to $\mathrm{Frg}_{i,j}$ to the root:

(i) The fragment at height u has been received correctly.

(ii) Otherwise, at least m_u out of the remaining $n_u - 1$ fragments have been received correctly.

The probability that at least one of the above two conditions is satisfied for height u is $(1 - p\sum_{v=0}^{m_u-1}\binom{n_u-1}{v}(1-p)^v p^{n_u-1-v})$. Therefore, the probability $f(i)$ that $\mathrm{Frg}_{i,j}$ is immediately verifiable upon receipt is $\prod_{u=i+1}^{h}\left\{1 - p\sum_{v=0}^{m_u-1}\binom{n_u-1}{v}(1-p)^v p^{n_u-1-v}\right\}$. □

Theorem 3. *The expected probability that any fragment in the proposed hash tree is immediately verifiable upon receipt is* $\sum_{i=0}^{h} f(i)(n_i/N)$ *(see Notations for the meanings of* h, n_i, *and* N*).*

Proof. Let X be a random variable representing the probability that each fragment from the proposed hash tree is immediately verifiable upon receipt. Then, the set of all possible values that X can take is $\{f(i) \mid 0 \le i \le h\}$. The probability $\Pr(X = f(i))$ is n_i/N. Therefore, the expected probability that any fragment in the proposed hash tree is immediately verifiable upon receipt is $\sum_{i=0}^{h} f(i)(n_i/N)$. □

Based on the derived formulas, we compared our approach with FIGOA in terms of the number of fragments and the immediate verification probability. Figures 5(a) and 5(b) show the results when varying the content size, fragment size, and fragment loss rate, respectively. To investigate the impact of the fragment loss rate, we used four different loss rates: $p = 0.02$ (2%), 0.04 (4%), 0.08 (8%), and 0.16 (16%). We set $|F(\cdot)| = 32$ and $r = 0.2$ for all of the test cases. To summarize the results, our approach provides a 52% higher immediate verification probability than FIGOA on average, while requiring 14% more fragments. Moreover, our approach achieved a high average immediate verification probability of 98.2% over all the test cases.

Effects of Buffer Replacement Policies. Because receivers have a fragment buffer of limited size, buffer replacement policies may affect the performance. The following are buffer replacement policies when $\mathrm{Frg}_{i,j}$ has been received under the condition that no empty slot is available in the buffer. We try to find a slot to replace with $\mathrm{Frg}_{i,j}$ in the following order of the policies described below:

(i) *Replacing unnecessary fragments (RUNF)*: if we have all m_k original fragments ($\mathrm{Frg}_{k,0}\sim\mathrm{Frg}_{k,m_k-1}$) for any height k, then we no longer need to keep any redundant fragment ($\mathrm{Frg}_{k,m_k}\sim\mathrm{Frg}_{k,n_k-1}$) of height k. In addition, any fragment at upper heights ($>k$) is no longer required because we have finished using the hash values included in those fragments. Thus we replace such a fragment in the buffer with $\mathrm{Frg}_{i,j}$.

(ii) *Replacing unverified fragments (RUVF)*: if any unverified fragment exists in the buffer, we replace it with $\mathrm{Frg}_{i,j}$ because it can sometimes be inauthentic one.

(iii) *Replacing fragments that will be less utilized (RFLU)*: if we fail to find any slot by following RUNF and RUVF, then we prefer to pick any slot with a fragment that belongs to any lower height ($<i$) but is not a descendant of $\mathrm{Frg}_{i,j}$. If there is no such a slot, then we search for fragments at the same height as $\mathrm{Frg}_{i,j}$ in the buffer, pick the one with the largest number of child fragments that have been received, and replace it with $\mathrm{Frg}_{i,j}$.

To investigate the impact of the above policies on immediate verification of fragments, we performed simulation studies of the following four strategies. In *strategy 1*, we applied all three policies (RUNF, RUVF, and RFLU) sequentially. In *strategy 2*, we applied the first two policies (RUNF and RFLU) only; if there is no buffer slot that is empty or occupied with unverified fragments, we pick a random slot and replace it with $\mathrm{Frg}_{i,j}$. In *strategy 3*, we applied RUNF only; if there is no buffer slot that is empty or occupied with unverified fragments, we pick a random slot and replace it with $\mathrm{Frg}_{i,j}$. In *strategy 4*, we did not apply any policies and pick a random slot and replace it with $\mathrm{Frg}_{i,j}$.

Figure 6 shows the simulation results. The x-axis shows the size of buffer, and the y-axis shows the ratio of the number of immediately verified fragments to the total number of received fragments. We used the following parameter settings: $|\mathrm{CO}| = 16\mathrm{K}$ bytes, $f = 700$ bytes, $|F(\cdot)| = 32$, $p = 0.16$, and $r = 0.2$. To investigate the impact of buffer size, we used four different buffer sizes: $25f$, $50f$, $75f$, and $100f$. Note that $25f$ $(= (m_0+m_1)f)$ is the minimum required buffer size according to RUNF. Among the four strategies, strategies 1, 2, and 3 (about 0.94) outperformed strategy 4 (about 0.91). Hence, it is our recommendation to use strategy 3 (RUNF only) because it can simplify the process of buffer replacement. Interestingly, as shown in Figure 6, the buffer size made little impact on the ratio of immediately verified fragments when the buffer size is sufficiently large (at least $25f$).

Resistance to DoS Attacks. Now we discuss the DoS resistance of our approach. The only strategy DoS attackers can take against our adaptive fragment forwarding mechanism is to inject a large number of immediately unverifiable fake fragments, hoping that they will be propagated over the network. Suppose that the victim node receives an unverifiable fake fragment from the attacker. According to our adaptive forwarding strategy, the fake fragment can be forwarded to the next node with a probability of $r_x = x/(x+y+z)$. However, as the victim continuously receives such unverifiable fake fragments in the DoS attack situation, the $r_z = z/(x + y + z)$ value for the victim increases, and this leads to a decrease in the value of $r_x (= 1 - r_z - r_y)$. That is, the probability that fake fragments can be propagated decreases. As a result, it is impossible for the attacker's fake fragments to be continuously forwarded to the next node. As an additional countermeasure, we can monitor the ratio of immediately unverifiable fragments received from each neighbor. If for any neighbor node the ratio is lower than the expected immediate verification probability, then we can suspect the neighbor to be abnormal.

Authenticity of Fragments. For a computationally bounded adversary running in a polynomial time, it is infeasible to generate fragments that can pass the verification process of our approach. The adversary could try to construct a hash tree over his own forged content and then digitally sign the hash value associated with the root node, pretending to be a trusted content publisher. Without the knowledge of the trusted publisher's private key, however, it is computationally infeasible to forge the trusted publisher's signature if we use a secure digital signature algorithm (e.g., RSA and ECDSA).

As an alternative strategy, given a fragment $\mathrm{Frg}_{0,j}$ of a valid content object from a trusted source, the attacker may attempt to find $\mathrm{Frg}'_{0,j}$ that satisfies $H(\mathrm{Frg}'_{0,j}) = H(\mathrm{Frg}_{0,j})$. As long as the attacker finds such $\mathrm{Frg}'_{0,j}$, it will pass our verification process while being misidentified as a fragment of the given content from the trusted source. However, since the cryptographic hash function $F(\cdot)$ used (e.g., SHA-2) is (weak) collision-resistant, it is also computationally infeasible to find such $\mathrm{Frg}'_{0,j}$ different from $\mathrm{Frg}_{0,j}$.

5. Related Work

There exist a number of studies on CCN-based vehicular networking. Several data naming approaches were developed for efficient data management and retrieval in CCN-based vehicular networks (e.g., [2, 6, 8, 15]). In particular, the authors in [6] proposed a naming scheme to describe a valid spatial and temporal scope of a given content. The authors in [15] proposed another naming scheme to indicate the popularity and cacheability properties of a given content. The authors in [8] developed a hybrid naming scheme that integrates hierarchical and flat naming.

CCN-based vehicular networking typically adopts interest and content flooding to enable fast content discovery, and thus it is critical to reduce the overhead introduced by flooding. In [10, 16], the authors proposed selective flooding approaches such that an intermediate node determines whether or not it floods the received interest according to its knowledge of any node having the requested content or the distance from itself to the known content provider. Several papers discussed randomizing and suppressing transmissions of interest and content messages to avoid potential collisions in shared wireless medium (e.g., [3, 9, 10, 16]). The authors in [7] suggested using cellular network for CCN signaling and short-range communication for content distribution, when multiple radio interfaces are available on network nodes.

Several other groups of researchers developed in-network caching and forwarding algorithms of contents, which are optimized for vehicular network environments (e.g., [6–10]). The key idea common to all these approaches is caching and forwarding overheard unsolicited contents as well as solicited contents in order to maximize the effectiveness of overhearing in wireless communication. In [6], based on the proposed naming scheme that represents a valid spatial and temporal scope of a content, the authors proposed an approach to selectively cache the contents within valid spatial and temporal ranges.

It is also crucial to protect CCN from malicious attacks. In particular, several groups of researchers proposed techniques to deal with illegal contents in CCN environments. In [17, 18], the authors developed a naming scheme to include the hash value of a content in the content name, so that an intermediate node can identify an invalid content by comparing the hash value of the received content with the hash value in the requested content name. In [19, 20], the hash value of the public key of a content publisher is included in an interest message in order for the receiver to check that the received content is from the actual owner of the public key. Several approaches were developed to take advantage of users' feedback of data authenticity in order to filter out invalid contents (e.g., [18, 21, 22]).

Several other approaches were developed to protect in-network caches from malicious attacks. In [23], an intermediate node probabilistically selects some of the received contents, verifies the selected contents, and caches only the correct contents. In [24], the authors proposed selective caching of contents according to their popularity in order to prevent cache pollution attacks that aim to intentionally replace popular contents currently stored in the caches with unpopular contents to degrade the effectiveness of in-network caching. In [25], the authors proposed a method to identify cache pollution attacks by analyzing incoming interests.

Finally, we provide an overview of erasure codes. There are two categories of erasure codes: *fixed rate* and *rateless*. Given m original data blocks, a fixed rate erasure code generates a limited number (n) of encoded blocks, where n should be determined prior to encoding, and the recovery of the original data blocks is guaranteed if a certain number

(m) of encoded blocks are given. Besides Rabin's IDA [12], Reed-Solomon code [26], Low-Density Parity-Check code [27], and Tornado code [28] belong to this category. On the other hand, a rateless erasure code can generate a potentially unlimited number of encoded blocks on demand, and if a certain number of encoded blocks are given, the original data blocks can be recovered with a certain probability, where the probability increases as the number of given encoded blocks increases. Luby Transform code [29] and Raptor code [30] are examples of rateless erasure codes.

In our approach, we used Rabin's IDA because of the following requirements. Our approach requires that generating all the encoded fragments in the proposed hash tree should be completed before the content publisher signs the root node of the hash tree in order to accommodate the immediate verification of encoded fragments. In addition, the recovery of any missing original fragment at any height i of the hash tree should be guaranteed if at least m_i encoded fragments out of the total of n_i fragments are received. Rabin's IDA meets these two requirements.

6. Conclusion

In this paper, we presented the design and analysis of a secure and DoS-resilient fragment authentication technique for CCN-based vehicular networks. Our approach not only guarantees the authenticity of fragments but also provides high resistance to DoS attacks through the immediate verification of incoming fragments and the adaptive forwarding strategy for immediately unverifiable fragments. The results of our analysis showed that the proposed approach provides much stronger security than FIGOA without imposing a significant overhead. Specifically, our approach achieves a high immediate verification probability of 98.2% on average over all the test cases in the experiments. In comparison with FIGOA, our approach also achieves on average 52% higher immediate verification probability, while requiring 14% more fragments than FIGOA.

Notations

CO:	Content object to be fragmented and authenticated
h, d:	Height and degree, respectively, of a hash tree
$\mathrm{Frg}_{i,j}$:	The jth fragment at height i of a hash tree
m_i:	Number of original fragments at height i before applying IDA
n_i:	Number of encoded fragments at height i after applying IDA to m_i original fragments ($n_i \geq m_i$)
r:	Redundancy rate of IDA ($r = (n_i - m_i)/n_i$)
N:	Total number of fragments generated from a hash tree
f:	Size of each fragment
$F(\cdot)$:	Cryptographic hash function (e.g., SHA-256)
SIG:	Digital signature of an input hash value
$M \parallel M'$:	Concatenation of M and M'
$\lvert M\rvert$:	Size of M in bytes
H_i:	Equal to $F(\mathrm{Frg}_{i,0}) \parallel \cdots \parallel F(\mathrm{Frg}_{i,n_i-1})$
$\mathbb{B}$:	Buffer to store received fragments
$\mathbb{V}$:	Set of authentic fragments in $\mathbb{B}$ ($\mathbb{V} \subseteq \mathbb{B}$)
$\mathbb{V}_i$:	Set of height i fragments in $\mathbb{V}$ ($\mathbb{V}_i \subseteq \mathbb{V}$)
$\mathbb{W}$:	Set of unverifiable fragments in $\mathbb{B}$ whose authenticity cannot be verified immediately upon receipt ($\mathbb{W} \subseteq \mathbb{B}$)
x, y, z:	Respective numbers of authentic, inauthentic, and unverifiable fragments received for a period of time
p:	Fragment loss rate.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2016R1A6A3A11930593) and by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2017-2015-0-00403) supervised by the IITP (Institute for Information & communications Technology Promotion).

References

[1] G. Grassi, D. Pesavento, G. Pau, R. Vuyyuru, R. Wakikawa, and L. Zhang, "VANET via named data networking," in *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS '14)*, pp. 410–415, Toronto, Canada, April 2014.

[2] Z. Yan, S. Zeadally, and Y.-J. Park, "A novel vehicular information network architecture based on named data networking (NDN)," *IEEE Internet of Things Journal*, vol. 1, no. 6, pp. 525–532, 2014.

[3] M. Amadeo, C. Campolo, and A. Molinaro, "CRoWN: Content-centric networking in vehicular ad hoc networks," *IEEE Communications Letters*, vol. 16, no. 9, pp. 1380–1383, 2012.

[4] 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 ACM Conference on Emerging Networking Experiments and Technologies (CoNEXT '09)*, pp. 1–12, December 2009.

[5] C. Ghali, A. Narayanan, D. Oran, G. Tsudik, and C. A. Wood, "Secure fragmentation for content-centric networks," in *Proceedings of the IEEE 14th International Symposium on Network Computing and Applications (NCA '15)*, pp. 47–56, Cambridge, Mass, USA, September 2015.

[6] L. Wang, R. Wakikawa, R. Kuntz, R. Vuyyuru, and L. Zhang, "Data naming in vehicle-to-vehicle communications," in *Proceedings of the IEEE Workshop on Emerging Design Choice in Name-Oriented Networking Workshop (INFOCOM '12)*, pp. 328–333, Orlando, Fla, USA, March 2012.

[7] A. Bazzi, B. M. Masini, A. Zanella, C. De Castro, C. Raffaelli, and O. Andrisano, "Cellular aided vehicular named data networking," in *Proceedings of the 3rd International Conference on Connected Vehicles and Expo, ICCVE 2014*, pp. 747–752, Austria, November 2014.

[8] W. Quan, C. Xu, J. Guan, H. Zhang, and L. A. Grieco, "Social cooperation for information-centric multimedia streaming in highway VANETs," in *Proceedings of the 15th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks, WoWMoM 2014*, aus.

[9] L. Wang, A. Afanasyev, R. Kuntz, R. Vuyyuru, R. Wakikawa, and L. Zhang, "Rapid traffic information dissemination using named data," in *Proceedings of the 1st ACM Workshop on Emerging Name-Oriented Mobile Networking Design—Architecture, Algorithms, and Applications (NoM '12)*, pp. 7–12, Hilton Head, SC, USA, June 2012.

[10] Y.-T. Yu, Y. Li, X. Ma, W. Shang, M. Y. Sanadidi, and M. Gerla, "Scalable opportunistic VANET content routing with encounter information," in *Proceedings of the 2013 21st IEEE International Conference on Network Protocols, ICNP 2013*, Germany, October 2013.

[11] 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.

[12] M. O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," *Journal of the ACM*, vol. 36, no. 2, pp. 335–348, 1989.

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

[14] L. Zhang and et al., *Named data networking (NDN) project*, NDN-0001, Xerox Palo Alto Research Center, 2010.

[15] Y.-T. Yu, M. Gerla, and M. Y. Sanadidi, "Scalable VANET content routing using hierarchical bloom filters," *Wireless Communications and Mobile Computing*, vol. 15, no. 6, pp. 1001–1014, 2015.

[16] M. Amadeo, C. Campolo, and A. Molinaro, "Enhancing content-centric networking for vehicular environments," *Computer Networks*, vol. 57, no. 16, pp. 3222–3234, 2013.

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

[18] C. Ghali, G. Tsudik, and E. Uzun, "Needle in a Haystack: Mitigating Content Poisoning in Named-Data Networking," in *Proceedings of the Workshop on Security of Emerging Networking Technologies*, San Diego, CA, USA.

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

[20] C. Ghali, G. Tsudik, and E. Uzun, *Elements of trust in named-data networking*, 2014, arXiv:1402.3332 [cs.NI].

[21] S. DiBenedetto and C. Papadopoulos, "Mitigating poisoned content with forwarding strategy," in *Proceedings of the IEEE INFOCOM 2016 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 164–169, San Francisco, CA, USA, April 2016.

[22] A. Compagno, M. Conti, C. Ghali, and G. Tsudik, "To NACK or not to NACK? Negative acknowledgments in information-centric networking," in *Proceedings of the 24th International Conference on Computer Communications and Networks, ICCCN 2015*, usa, August 2015.

[23] G. Bianchi, A. Detti, A. Caponi, and N. Blefari-Melazzi, "Check before storing: What is the performance price of content integrity verification in LRU caching?" pp. 60–67.

[24] M. Xie, I. Widjaja, and H. Wang, "Enhancing cache robustness for content-centric networking," in *Proceedings of the IEEE Conference on Computer Communications, INFOCOM 2012*, pp. 2426–2434, USA, March 2012.

[25] 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.

[26] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the Society For Industrial & Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[27] R. G. Gallager, "Low-Density Parity-Check Codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[28] M. G. Luby, M. Mitzenmacher, M. . Shokrollahi, and D. A. Spielman, "Efficient erasure correcting codes," *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, vol. 47, no. 2, pp. 569–584, 2001.

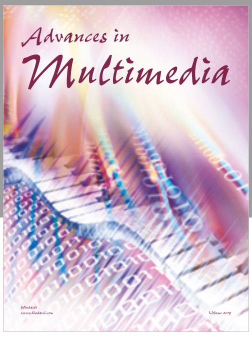
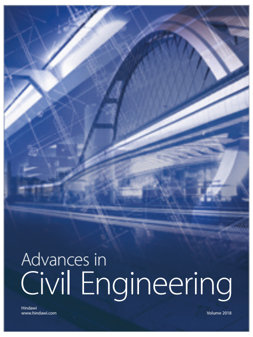
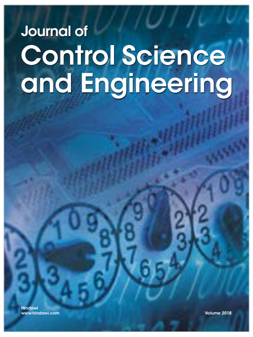
[29] M. Luby, "LT codes," in *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science*, pp. 271–280, IEEE, Vancouver, Canada, November 2002.

[30] A. Shokrollahi, "Raptor codes," *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, vol. 52, no. 6, pp. 2551–2567, 2006.

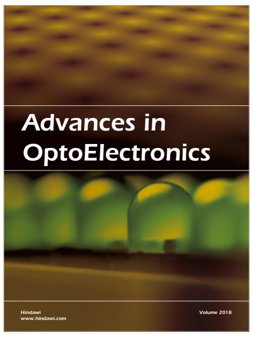
Journal of
Engineering

The Scientific World Journal

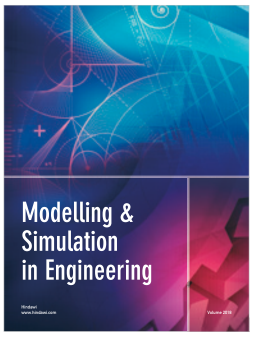
International Journal of
Rotating Machinery

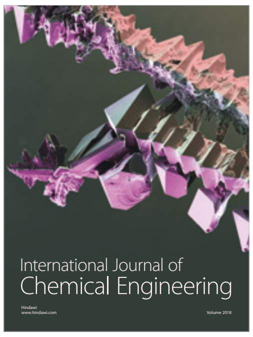
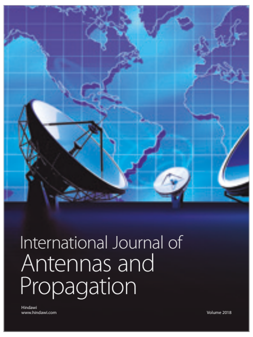
Journal of
Sensors

Advances in
Multimedia

Advances in
Civil Engineering

Journal of
Control Science and Engineering

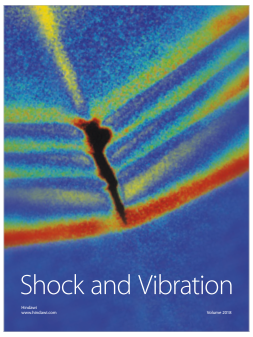
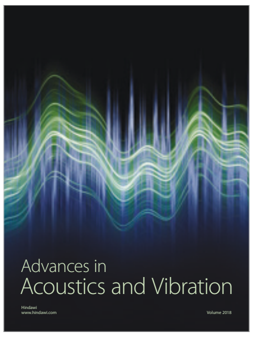
Journal of
Robotics

Journal of
Electrical and Computer Engineering

Advances in
OptoElectronics

VLSI Design

International Journal of
Navigation and Observation

Modelling & Simulation in Engineering

International Journal of
Aerospace Engineering

International Journal of
Chemical Engineering

International Journal of
Antennas and Propagation

Active and Passive Electronic Components

Shock and Vibration

Advances in
Acoustics and Vibration