Hindawi
Wireless Communications and Mobile Computing
Volume 2022, Article ID 4466942, 16 pages
https://doi.org/10.1155/2022/4466942

Research Article

A Secure and Cached-Enabled NDN Forwarding Plane Based on Programmable Switches

Ningchun Liu,[1] **Shuai Gao**,[1] **Lei Yu**,[1] **and Guobiao He**[2]

[1]*School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing, China*
[2]*National Computer Network Emergency Response Technical Team/Coordination Center of China (CNCERT/CC), Beijing, China*

Correspondence should be addressed to Shuai Gao; shgao@bjtu.edu.cn

Received 3 June 2022; Accepted 28 September 2022; Published 3 November 2022

Academic Editor: Haitao Xu

Recently, the rapid development of software-defined networking (SDN) and programming protocol-independent packet processors (P4) provides a potential possibility for the deployment of Named Data Networking (NDN), which has aroused tremendous attention in academia. Existing P4-based NDN solutions mainly focus on how to describe the stateful forwarding characteristics of NDN in a programmable switch environment. However, the existing solutions still face many challenges such as cache availability and data confidentiality and do not support retransmission of interest packets and multicast forwarding of data packets. In this paper, we propose a new NDN forwarding plane based on programmable switches to address the above challenges. We design a decoupled cache module to avoid a large impact on the data plane forwarding performance when the cache function is enabled. Also, we enhance the design of the existing P4-based NDN forwarding plane to support interest retransmission and multicast forwarding of data packets. In addition, with the advantage of network programmability of P4 technology, we extend the content permutation algorithm and integrate it into the NDN forwarding plane, which makes our scheme support lightweight secure forwarding. Finally, we evaluate our scheme in the prototype system and conduct comparative experiments with representative schemes. Experiment results show that our scheme outperforms it in terms of content retrieval latency and received throughput and can support lightweight secure forwarding with low cost.

1. Introduction

With the rapid development of network technology, content services represented by online videos have gradually become the main services of today's Internet. According to data provided by the Cisco Visual Networking Index, video traffic will account for 82 percent of IP traffic by 2022 [1]. In this context, the traditional TCP/IP network adopts a host-centric communication mode and faces many challenges such as redundant transmission. Furthermore, users are more concerned about the content itself, rather than the location of the content provider. Since 1999, a series of ICN architectures have been proposed. ICN adopts a content-centric communication model and decouples the location of content and content providers. In the ICN, content copies can be cached at intermediate routing nodes in response to subsequent requests for the same. Because it has the characteristics of ubiquitous caching in the network, it is considered to help reduce redundant transmission in the network [2].

As the most promising project in the ICN architecture, NDN follows a content-centric design philosophy and addresses and routes based on named data at the network layer. The forwarding based on named data in NDN is fundamentally different from the packet processing logic of the traditional TCP/IP network, and the existing software-based NDN routers greatly limit its forwarding performance, which makes NDN encounter challenges in large-scale deployment [3]. The emergence of software-defined networking (SDN) and programming protocol-independent packet processors (P4) provides a potential possibility for constructing high-performance NDN routers and deploying NDN on a large scale, which has attracted widespread attention in the academic community.

Recently, some researchers have carried out some works on P4-based NDN [3–6], which mainly focus on using P4 to

describe the stateful forwarding characteristics of NDN. Specifically, these works explore how to implement the parsing of hierarchical content names and three data structures in forwarding model of NDN, including forwarding information base (FIB), content store (CS), and pending interest table (PIT). However, the initial goal of the P4 design is protocol-independent forwarding. Since there is no special design for in-network caching, how to use P4 to describe the characteristics of NDN in-network caching has become the challenge faced by researchers. Existing works either use the register [7] and queue [8] of the programmable switch or implement the conceptual logic of content storage [4], which has the problem of small cache space and poor availability. In addition, although NDN adopts a content-centric security model and can protect the security of data by encrypting data at the application layer, it is necessary to carry out a corresponding security design for each application with security requirements. Compared with traditional NDN data encryption at the application layer, P4 provides the potential possibility for NDN network layer data encryption.

In this paper, we propose a new NDN architecture with a secure and cached-enabled NDN forwarding plane based on programmable switches. In this architecture, a decoupled cache module is presented to avoid a large impact on the data plane forwarding performance, where the CS module is decoupled into the CS-list and the CS-server. The bloom filter-based CS-list enables that only interest packets being satisfied by the local cache can be forwarded to the CS-server. By extending the PIT in the existing P4-based NDN solutions, our scheme support the retransmission of interest packets. Besides, to support multicast forwarding feature of data packets, we add a data clone operation to the forwarding pipeline, which significantly reduces the congestion of data packets in the programmable switch and improves the network throughput. In addition, our scheme also supports secure forwarding to mitigate the threat of eavesdropping attacks. The secure forwarding can be divided into two phases. First, the control plane sends a security configuration to a group of programmable switches. The security configuration mainly consists of symmetric keys written into registers on programmable switches. Second, according to the requirements of service, programmable switches in data plane can encrypt and decrypt security-sensitive traffic against eavesdropping attacks combined with content permutation.

Finally, we build the prototype system based on the software programmable switches, which realizes the basic functions of NDN, especially the high-availability in-network caching function and lightweight secure forwarding function proposed in this paper. Next, we conduct comparative experiments to evaluate the performance of our mechanism with representative AES-based scheme, S-BOX matrix-based scheme, and NDN.p4 solution. Experiment results show that our scheme outperforms in terms of content retrieval latency and received throughput. Besides, our scheme can support on-demand secure forwarding with low cost.

Our main contributions can be summarized as follows:

(i) We propose a new NDN architecture with a secure and cached-enabled NDN forwarding plane based on programmable switches, which can support high-availability in-network caching and lightweight securing forwarding

(ii) We present a decoupled cache module in the architecture, which can provide high-availability in-network caching capability for the data plane without affecting the forwarding performance as much as possible

(iii) We enhance the design of the PIT and egress pipeline in traditional NDN forwarding plane to support retransmission of interest packets and multicast forwarding of data packets

(iv) We integrate the extended content permutation algorithm into the NDN forwarding plane. The programmable switch can encrypt and decrypt data packets with security requirements against eavesdropping attacks

The rest of this paper is organized as follows: Section 2 surveys the related work. Section 3 introduces the example scenario. Section 4 presents the proposed architecture. In Section 5, we describe the design of the secure and stateful forwarding plane. Several experiments based on prototype system are done in Section 6 to evaluate the performance. Security analysis is discussed in Section 7. Finally, Section 8 concludes this paper.

2. Related Work

This section reviews the related work on traditional schemes for protecting data confidentiality in NDN and P4-based schemes for NDN and securing forwarding.

2.1. P4-Based NDN Solutions. In 2016, Signorello et al. [3] preliminarily implemented an NDN router for the first time by using P4, which contains most of the functions of NDN. Due to the limitations of stateful storage in $P4_{14}$, the implementation can not support cache requests. On this basis, Miguel et al. [9] presented an extended design of an NDN router in $P4_{16}$, which especially supports cache requests implemented with P4 externs. Due to the reference software target, BMv2-ss, having little support for it, the implementation is at the software logic level. Hou et al. [6] proposed a new P4-based NDN solution in which the cache function is supported by using an NFD-based cache server. Due to the use of a separate cache server, it is difficult to fully demonstrate the performance improvement brought by the cache. In addition, the security issues brought by data plane eavesdropping are not considered in this architecture. Karrakchou et al. [10] designed an enhanced NDN architecture by using P4, which extends the design of the traditional NDN forwarding pipeline and the existing FIB and PIT tables to provide support for several content delivery patterns. In addition, the architecture supports the execution of multiple P4 forwarding functions. Guo et al. [5] proposed an NDN routing mechanism in the P4 environment based on the NLSR protocol, which supports resource-location management and routing calculation. However, these

mechanisms [5, 10] do not take the functionality and performance of caching into account, which is an important feature for NDN.

In addition, the existing P4-based NDN schemes pay little attention to the security of the network layer. Compared with traditional NDN data encryption at the application layer, P4 provides the potential for NDN network layer data encryption.

2.2. Traditional Access Control Schemes in NDN. Different from the channel-based security model in traditional TCP/IP networks, NDN adopts the content-based security model. To enhance the confidentiality of data, researchers have designed many data access control schemes in NDN. Due to space limitations, we review the relevant work in the past five years. Other works can refer to the survey [11].

Xue et al. [12] designed a secure, efficient, and accountable edge-based access control framework for NDN, which adopts group signature to achieve anonymous authentication performed at the network edge. In addition, this framework uses the hash chain technique to reduce the overhead when users make continuous requests for the same file. However, the data is still transmitted in plain text in the network, and it is difficult to guarantee the confidentiality of the data.

Zhang et al. [13] presented a name-based access control scheme, which ensures data confidentiality in NDN. Combined with attributed-based encryption, the scheme supports fine-grained access control policies. By leveraging extended NDN naming conventions, the scheme also automates cryptographic key management. Misra et al. [14] proposed a data access control scheme based on broadcast encryption. In this scheme, even if the authentication entity is offline, the content producer can still allow legitimate consumers to access the data. Ning-Chun et al. [15] designed a data access control mechanism, which introduces the precalculated and cached auxiliary key block to reduce the retrieval delay of the auxiliary key block and the decryption cost for consumers. Besides, combining a two-dimensional one-way function, the mechanism ensures the uniqueness of the consumer's secret share.

The above-mentioned schemes [13–15] are conceptually similar. In these schemes, content producers encrypt their content before distributing them to the network. Consumers need to authenticate themselves and obtain the content decryption keys to be able to decrypt and access the content. Considering that the overhead of a symmetric encryption algorithm is much smaller than that of an asymmetric encryption algorithm, the content is usually encrypted using a symmetric encryption algorithm, and the symmetric key (also called the content key) is encrypted using an asymmetric encryption algorithm. However, the security cost is still a challenge for resource-constrained edge network scenarios. Also, most of these schemes are deployed at the application layer, which can support encryption and decryption of application data with security requirements by an extra specific development. Obviously, deployment can become cumbersome for multiple applications with security requirements.

2.3. P4-Based Securing Forwarding and Cache Solutions. For P4-based securing forwarding, Oliveira et al. [16] introduced traditional AES algorithm into programmable data plane, which allows nodes in data plane to establish secure channels between each other. However, this encryption mechanism still imposes high computational costs in the data plane, due to the interaction process including multiple key agreements between nodes. Liu et al. [17] proposed a P4-based network immune scheme (P4NIS) against the eavesdropping attacks, which is equipped with three lines of defenses. In the third line, the scheme encrypts all packet payloads in the application layer using traditional cryptographic mechanisms. In addition, the scheme reencrypts all packet headers in the transport layer using S-BOX matrix-based cryptographic schemes in the second line. However, the operation of S-BOX matrix-based cryptographic schemes is cumbersome and cannot meet the requirements of encoding and decoding data packets at line rates. Lin et al. [18, 19] designed an enhanced content permutation algorithm (eCPA) with programmable switches that can ensure the confidentiality of data in the ramable data plane. eCPA can perform encoding and decoding operations of data packets at line rate. Nevertheless, since NDN adopts a content-centric communication mode, eCPA cannot be directly integrated into P4-based NDN architecture.

In terms of P4-based cache, Jin et al. [7] used the software switch register unit to achieve cache, which has the best performance on P4. The size of the register unit of the software switch is limited. It is impractical to support packet buffering in MTU 1500 scenarios without modifying the source code. Qu et al. [8] proposed a P4-based queue cache. Queue cache refers to the use of a virtual port to store packets. When the packet leaves the queue, by retrieving the packet, the packet enters the queue again to realize packet cache. However, the cache capacity required by NDN is larger than the capacity provided by the above solutions.

3. Example Scenario

To facilitate an understanding the rest of this paper, we present a typical SDN-based battlefield application scenario in Figure 1. In this scenario, there are four types of entities: (i) command center ("/military/task/control") that sends control commands, (ii) battle units ("/military/task/uav" and "/military/task/squad(1/2)") that receive commands from the command center, (iii) forwarders (Programmable Switch(1/2) and Gateway) that forward packets among battle units and command center, and (iv) SDN controller that drives the secure forwarding of the forwarders by configuring the flow table.

This battlefield scenario ensures that eavesdropping attackers cannot obtain certain pieces of data sent by the command center to the combat units. As an example, when the command center sends a command intended to squad 1, all the other entities or attackers in the wide area network (WAN) should not be able to see the content even if they have retrieved the data packets. In the rest of the paper, we

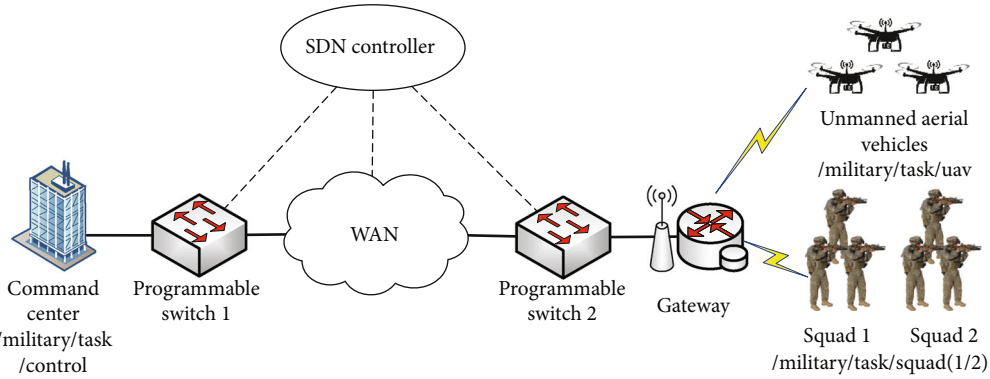


Figure 1: A SDN-based battlefield communication scenario.

illustrate how our architecture guarantees the confidentiality of communications.

4. Requirements, Proposed Architecture, and Security Assumptions

4.1. Requirements. We fully analyze the requirements for current P4-based NDN architecture as follows.

4.1.1. Caching Availability. Due to in-network caching being one of the important features of NDN, supporting high-availability caching is necessary for the proposed architecture. Specifically, the programmable data plane should support large-capacity caching. In addition, the support of the in-network caching feature should not affect the forwarding performance of the programmable data plane for other packets as much as possible.

4.1.2. Data Confidentiality. To mitigate eavesdropping attacks against packets, the proposed architecture should protect the confidentiality of data. Furthermore, the protection for data confidentiality should be lightweight in edge network scenarios.

4.2. Proposed Architecture. To meet the above requirements, we present a new P4-based NDN architecture, which supports more practical caching functionality and lightweight confidentiality protection of data at the network layer. Following the principles of SDN, the proposed architecture separates the control plane and the programmable data plane. The controller uses the control link between the control plane and the programmable data plane to configure the programmable switches. Figure 2 shows an overview of the proposed architecture.

The control plane is designed for topology discovery and management and data plane configuration containing table configuration, security configuration, and caching management. More specifically, the controller perceives the network topology through the link layer discovery protocol, including programmable switches and communication links. The controller also provides a content name registration service to establish resource-location mapping and store it in the database. The information in the database will subsequently be converted into configuration parameters for the programmable switches in the programmable data plane. Besides, the controller needs to set the cache policy and configure the security parameters of programmable switches.

In the programmable data plane, programmable switches focus on the packet forwarding for interest and data packets sent by consumers and producers, which mainly contains parser, PIT and FIB module, CS module, and encryption and decryption module. Since the NDN packets adopt the type-length-value (TLV) format, the traditional parser module needs to be extended to parse the variable-length NDN packet. Besides, according to the security requirements of the application, the encryption and decryption module performs encryption and decryption operations on the corresponding NDN packets. In ingress pipeline and egress pipeline, traditional PIT, FIB, and CS functionalities are included. In order to adapt to the environment of programmable switches, we split the traditional CS into CS-list and CS-server to improve caching performance. In addition, the CS module is integrated into the programmable switches via the virtual face. The design details of the programmable data plane are presented in Section 5.

4.3. Security Assumptions. We elaborate on the security assumptions of the proposed architecture as follows:

(i) The controller configures the programmable switches by using the control link between the control plane and the programmable data plane, which can usually be encrypted with TLS/SSL. We assume that the control link is secure; that is, the information transmitted in the control link cannot be eavesdropped on and tampered with

(ii) We assume that there are potential attackers in the network, and the attackers may launch man-in-the-middle attacks or eavesdropping attacks. In the man-in-the-middle attack, the attacker tampers with the content information of interest and data packets. In the eavesdropping attack, the attacker eavesdrops on the content information

(iii) Considering that the designed architecture is mainly to mitigate man-in-the-middle attacks or

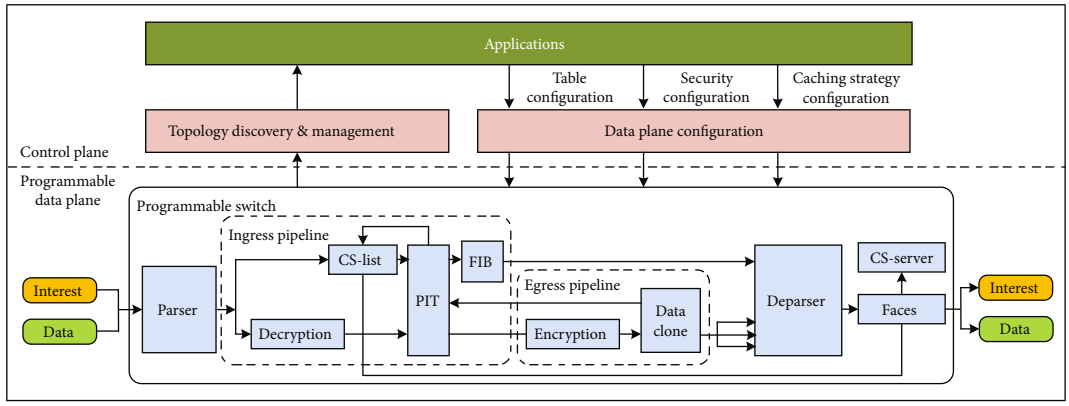


Figure 2: Overview of the architecture.

eavesdropping attacks in the network, we assume that consumers and content producers are trusted in the proposed architecture

5. Programmable Data Plane Design

Based on the previous work, we enhance the design of the programmable data plane, which support high-availability in-network caching, lightweight secure forwarding against eavesdropping attacks, and packet retransmission. In addition, the native multicasting function of NDN is realized in the proposed scheme.

In this section, we introduce the programmable data plane design from the following four main parts: (1) parser module, (2) PIT and FIB module, (3) CS module, and (4) encryption and decryption module. Finally, we show the workflow of our mechanism. It should be noted that PIT and FIB module, CS module, and encryption and decryption module are all main components of ingress pipeline and egress pipeline.

5.1. Parser Module. The main function of the parser module is to parse the header field of the packet from the outside to the inside according to the parsing sequence specified by the P4 target. The parsing process of each header field is called a parser method. When the parser module is working, after performing a parser method, the packet will enter the next parser method according to the byte offset of the currently processed header field and the next pending header field specified by the P4 target.

Since the NDN packet is composed of multiple continuous TLV blocks, we use the header stack data structure to simplify this series of the same type of adjacent header parsing. For continuous TLV blocks, P4 target will repeat the TLV header parsing process several times until all TLV block parsing is completed.

In addition, we extend the parsing process of the parser module to support version discovery of the NDN protocol and parsing of three types of NDN packets, including interest packet (0×05), data packet (0×06), and NACK packet ($0 \times 0a$).

5.2. PIT and FIB Module. Firstly, the variable name is converted into a fixed-length hash string by the hash function to be stored, parsed, and matched. FIB and PIT of NDN match the routing interest packet by the longest prefix of the content name. All packets must be converted from name to hash to enter the forwarding process. Hash is stored in the metadata of P4 target, which is only used for internal processing of the switch. Hash index and PIT table stored in cache will not be sent at the port and will be released after the packet processing process.

In the programmable switch, PIT and CS-list are stored in the register, which is completely handled by the switch itself. FIB is flow table form and issued by the control plane. The advantage of this form is that unnecessary communication between the control plane and the data plane is minimized to meet the decoupling requirements of the control plane and the data plane.

Then, as shown in Figure 3, different processing is performed depending on the type of the packet.

5.2.1. NACK Packet. When receiving an NACK packet, the programmable switch will forward it to the controller in the control plane for processing.

5.2.2. Interest Packet. When receiving an interest packet, the programmable switch will process it as follows:

Step 1. Match in the CS-list to determine whether there is a cache. The interest packet is directly forwarded to the face of CS-server when the cache is hit.

Step 2. Enter the PIT (recording the previously forwarded interest packets and their faces entering the switch) for matching. If the corresponding entry is found in the PIT, the routing node has forwarded the same requested interest packet but has no response yet. It is only necessary to discard the interest packet and update the face to the PIT. If the interest packet is received multiple times on the same face, it is regarded as a retransmitted packet, so the matching action table FIB is applied.

Step 3. If the corresponding matching table entry is not found in the PIT, then the content request is processed by

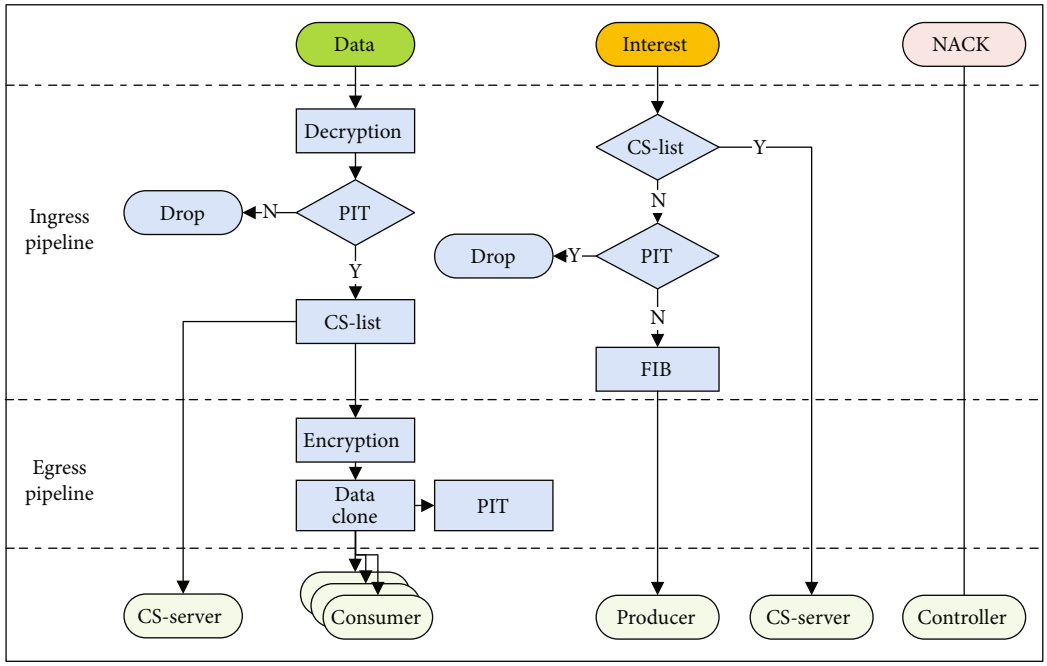


Figure 3: Packet processing flow in programmable switches.

the routing node for the first time and needs to be searched in the FIB (recording the forwarding rules between different nodes). The request is forwarded to other routing nodes according to the interface list corresponding to the content name in the FIB, and the PIT entry corresponding to the content name is modified to indicate to the next same content request that the interest packet of the content name is being requested to be resolved.

5.2.3. Data Packet. When receiving a data packet, the programmable switch will process it as follows:

Step 1. First, it is determined whether the data packet is an encrypted packet. If it is an encrypted data packet, decrypt the data packet. If it is not an encrypted data packet, transfer the data packet to PIT for subsequent matching operations. Date packets enter the PIT for matching. No matching entry means it is an unsolicited data packet and should be dropped.

Step 2. Next, match the data packet in PIT. If there is no hit, the data packet is unsolicited and should be dropped. If there is a hit in the PIT, the name of the data packet should be stored in the CS-list to wait for other consumer requests.

Step 3. Then, forward the data packet to consumers and the CS-server through faces according to the PIT. Before that, it is also necessary to encrypt and clone the data packet. If the data packet is a plain data packet, the data packet needs to be encrypted. In the data clone operation, the original data packet is first forwarded to the face with the smallest sequence number, and then, a loop is constructed through the clone operation in PIT. When a data packet is sent to each face, the corresponding bit in the PIT is set to zero. When the corresponding bits in the PIT are all zeros, the operation of sending data packets to the face stops. We specify the CS-server face to a specific value.

When the programmable switch performs the clone operation in the egress pipeline, a new thread is created to reexecute the egress process each time a new data packet is cloned. During this period, the switch can continue to process the newly arrived packets to make full use of face resources.

5.3. CS Module. Considering that in the traditional P4-based NDN solution, the CS is directly deployed in the pipeline of the programmable switch, which will have an impact on the forwarding performance of the programmable switch. In the proposed architecture, the decoupled cache module consists of two parts, CS-list and CS-server, as shown in Figure 4.

The CS-list is deployed in the registers and adopts a bloom filter-based structure, which only determines whether there is a cache, and does not perform a cache lookup operation. The CS-server is an independent process of the programmable switch, which will perform a name lookup operation on the received interest packets and also respond to the corresponding data packets.

In the decoupled cache module, the received interest packet will enter the CS-list based on the bloom filter for matching. If it hits, it will be forwarded to the CS-server for cache search and response. If there is no hit, the interest packet will enter the FIB for matching and forwarding. In CS-server, the data package corresponding to the interest package will be searched, and the data package will be returned along the reverse path. If the corresponding data packet is not found, the CS-server will return a NACK

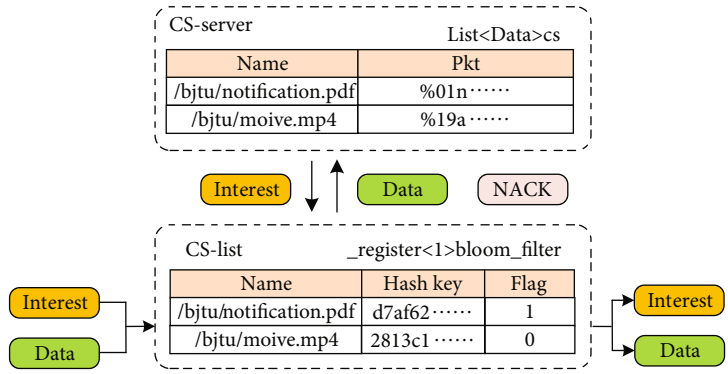


Figure 4: The schematic diagram of the cache module.

packet, which will be reencapsulated as an interest packet and sent to the FIB for matching and forwarding.

5.4. Encryption and Decryption Module. The encryption and decryption module encrypts and decrypts the payload of the data packets according to the security level field and flag bit in the header.

After receiving the data packets to be encrypted, the encryption module will process the data packets according to the procedure, as shown in Figure 5.

First, the payload of the packet is parsed as the header and divided into n blocks. Next, the n blocks will be secretly permuted according to the CPA-based encryption algorithm to complete the encryption process. Finally, the encryption module generates an encrypted packet.

The principle of CPA-based encryption algorithm is shown in Algorithm 1. For $n \geq 2$, we divide the payload field to be encrypted into n codewords. Denote n codewords for the i^{th} state as $R_n^i = \{r_1^i \| r_2^i \| \cdots \| r_n^i\} (i = 0)$, where r_n is the n^{th} codeword of the field. $R_n^i = \{r_1^i \| r_2^i \| \cdots \| r_n^i\} (i = n-1)$ represents the result after $n-1^{\text{th}}$ permutation. Denote a binary cypher of size $n-1$ as $K_{n-1} = \{x_1 \| x_2 \| \cdots \| x_{n-1}\}$, where $x_i \in \{0, 1\}$ for $1 \leq i \leq n-1$. The R_n^0 is permuted with the key K_{n-1}, and the output result is R_n^{n-1}. Besides, we use the Buffer to store the intermediate value in secret permutation. In the Algorithm 1, line 4 to line 14 are a loop that is executed $n-2$ times. We divide the encoding actions by key value. If $x_i = 1$, we shift code words from $\{r_1^i \| \cdots \| r_{i-1}^i \| r_i^i \| r_{i+1}^i \| \cdots \| r_{n-1}^i \| r_n^i\}$ to $\{r_1^i \| \cdots \| r_{i-1}^i \| r_n^i \| r_i^i \| r_{i+1}^i \| \cdots \| r_{n-1}^i\}$, and the latter is n codewords for the $i+1^{\text{th}}$ state. If $x_i = 0$, do not change the position of codeword r_i. If $i = n-1$, we execute lines 15 to 23. Finally, each codeword is disordered and located in a different position.

Since the operation of CPA-based decryption is opposite to the encryption operation, we will not elaborate further in this paper.

5.5. Workflow. The communication flow in the network is shown in Figure 6, which contains the system initialization phase and content retrieval phase. According to the different network entities responding to the interest packet, the content retrieval phase can be divided into two categories: content retrieval phase (from the producer) and content retrieval phase (from cache).

5.5.1. System Initialization. The control plane first generates a symmetric key and delivers the symmetric key to the programmable switch on the programmable data plane through a TLS-encrypted secure channel. After receiving the symmetric key, the programmable switch stores the symmetric key in its own register. In addition, the controller preconfigures the cache replacement policy in the programmable switch.

5.5.2. Content Retrieval. In the content retrieval phase, after the programmable switch 1 receives the interest packet carrying the security granularity requirement, it will first match its CS-list. If the CS-list hits, the interest packet will be sent to the CS-server through the face based on interprocess communication, and the CS-server will respond to the data packet corresponding to the interest packet. If there is no hit in the CS-list, the programmable switch will encrypt the interest packet according to the requirements of the security granularity. After that, the programmable switch sends the encrypted interest packet to the next-hop programmable switch 2.

After the programmable switch 2 receives the interest packet carrying the security granularity requirement, the switch first determines whether the interest packet is an encrypted interest packet according to the flag bit. If it is an encrypted interest packet, first perform the decryption operation and match its CS-list. If it is not an encrypted interest, it will directly match its CS-list. The execution operation in the CS-list is similar to that of the programmable switch 1, except that the encryption operation is no longer performed.

Contrary to the forwarding process of the interest packet, after the programmable switch 2 receives the responding data packet, the switch encrypts the data packet and forwards it to the programmable switch 1. After receiving the encrypted data packet, the programmable switch 1 decrypts the data packet and forwards the data packet to consumer 1.

6. Deployment and Performance Evaluation

In this section, we first implement the proposed scheme and build the prototype system based on it. Then, we evaluate the performance of the proposed scheme from four metrics.

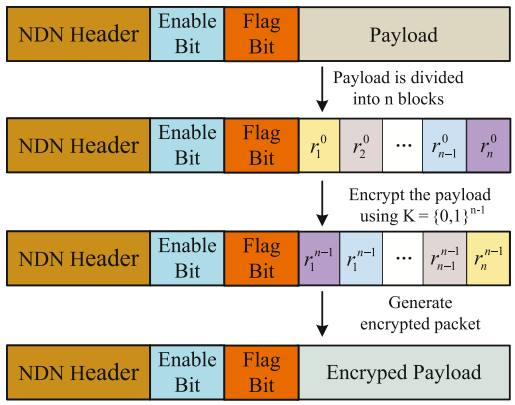


Figure 5: The encryption procedure for NDN packet.

Input:
RawPacket: $R_n^i = \{r_1^i \| r_2^i \| \cdots \| r_n^i\} (i = 0)$.
Output:
EncPacket: $R_n^i = \{r_1^i \| r_2^i \| \cdots \| r_n^i\} (i = n - 1)$.

```
1  //generate the key
2  key = KeyGenerate() = K_{n-1} = {x_1 || x_2 || ... || x_{n-1}};
3  //encrypt the RawPacket using the key K_{n-1}.
4  //step 1 to step n-2
5  for i = 1 to n − 2 do
6      if x_i = 1 then
7          Buffer = r_n^{i-1};
8          for k = n − 1 to i do
9              r_{k+1}^i = r_k^{i-1};
10         end
11         r_i^i = Buffer;
12     else
13         NoAction();
14     end
15 end
16 //step n-1
17 while i = n − 1 do
18     if x_i = 1 then
19         Buffer = r_n^{i-1};
20         r_{i+1}^i = r_i^{i-1};
21         r_i^i = Buffer;
22     else
23         NoAction();
24     end
25 end
```

Algorithm 1: The CPA-based encryption algorithm.

6.1. Deployment Environment. We implemented the proposed scheme by combining the simplified NDN Forwarding Daemon (NFD) [20] with the enhanced NDN.p4 [3] using libraries from the behavioral model (BMv2) software switch [21]. More specifically, the traditional NFD is simplified to only retain the function of content store, and NDN.p4 is extended to support packet retransmission and secure forwarding. In addition, interest/data packets are exchanged between NDN.p4 and NFD through inter-process communication. The version of the P4 language accepted by the proposed scheme's P4 function target is the standard $P4_{16}$ version.

The topology of the prototype system is shown in Figure 7, which contains two consumers (consumer 1 and consumer 2), two programmable switches (programmable switch 1 and programmable switch 2), and a content producer. To more realistically show the network environment when the content provider provides services, we add a SPIRENT network emulator to the prototype system, which is used to simulate the latency of the Internet. The producer is connected to the programmable switch 2 and serves two consumers connected to a programmable switch 1. The latency of the network emulator is set to obey the delay characteristics of Internet ranging from 2.5 ms to 5 ms. Table 1 provides a summary of device parameter information in the prototype system.

6.2. Experiment Result and Analysis. We evaluate the performance of the proposed scheme from the following metrics: (1) security cost, (2) content retrieval latency, (3) packet processing latency, and (4) received throughput.

(i) Security cost refers to the total time of encryption operation and decryption operation of interest/data packets

(ii) Content retrieval latency refers to the time interval between when consumer sends the first interest packet and receives the last corresponding data packet

(iii) Packet processing latency refers to the time of each blocks in the programmable switches

(iv) Received throughput refers to the real-time throughput of data packets received by consumers, which reflects the forwarding performance of programmable switches

6.2.1. Security Cost. We evaluate the encryption and decryption time of the proposed scheme, the representative AES-based scheme [16], and S-BOX matrix-based scheme [17]. In the comparative experiment, we set the payload length of the NDN data packet to 1024 bits, and the key lengths used are all 128 bits. Correspondingly, in the proposed scheme, the codeword is 8 bits.

As shown in Figure 8, the encryption and decryption time of the proposed scheme in the programmable switch has a decrease of 96.9% and 84%, respectively, compared to the AES-based scheme and S-BOX matrix-based scheme.

6.2.2. Content Retrieval Latency. To demonstrate the impact the proposed scheme with high-availability caching and lightweight on-demand secure forwarding functions on the forwarding performance of programmable switches, we first compare the content retrieval latency with and without in-network caching in different schemes. In our experiment, the consumer sends 10,000 interest packets at a constant rate where the interest packets carry names with four name

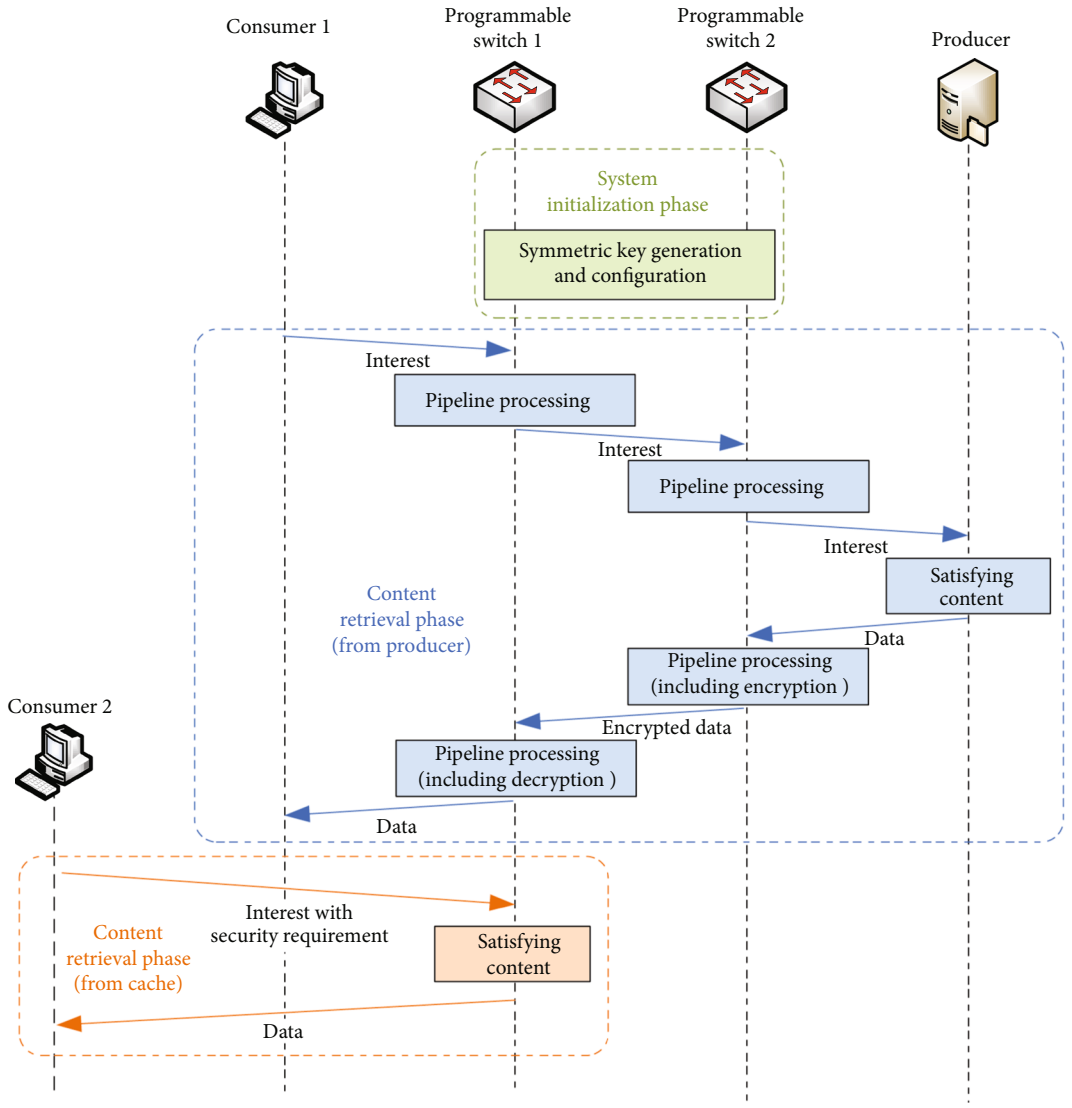


Figure 6: Communication flow in the network.

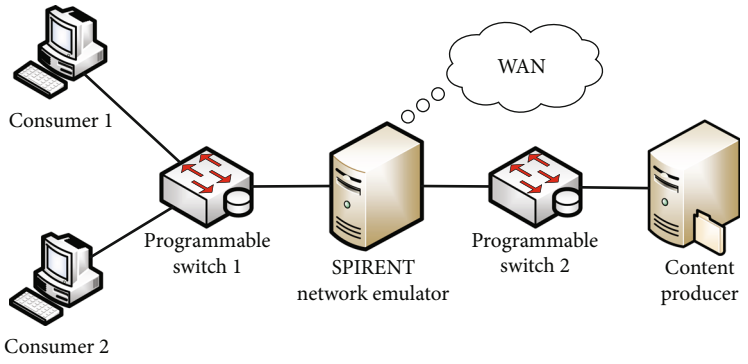


Figure 7: Topology of the prototype system.

components. In addition, the payload of the response data packet is set to 1024 bits.

Table 2 shows the content retrieval latency in different schemes, which are averaged over 10,000 experiments.

Compared with the traditional NDN.p4 solution [3], the proposed scheme is approximately equal in content retrieval latency when the in-network caching is disabled. When the in-network caching function is enabled, the proposed

Table 1: The summary of device parameter information in the prototype system.

Num	Device	Parameter	Count
1	BMv2 switch	Intel Xeon Gold 5218@2.30 GHz 64.0 GB RAM	2
2	Consumer	Intel Xeon E3-1230v2@3.30 GHz 4.0 GB RAM	2
3	Producer	Intel Xeon E5-2620@2.00 GHz 16.0 GB RAM	1
4	Network emulator	SPIRENT 2*M series V2 2*1G-copper (GEM mode)	1

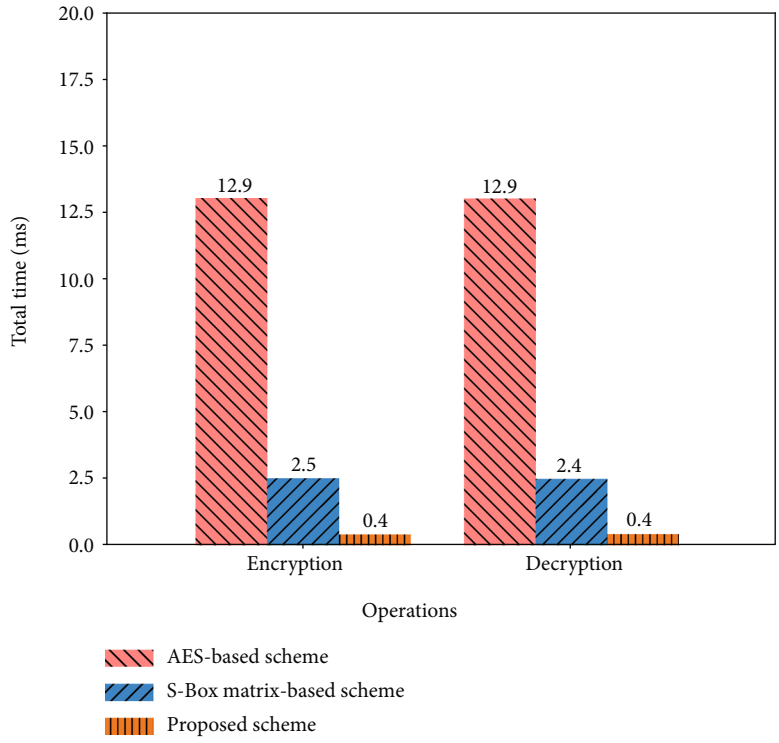


Figure 8: The comparison for encryption and decryption time.

scheme reduces the content retrieval latency by 57%. It is worth mentioning that the traditional NDN.p4 solution does not support the in-network caching function.

In addition, when the secure forwarding function is enabled, the content acquisition delay will increase by 0.8 ms (not supporting in-network caching) and 0.002 ms (supporting in-network caching), which is acceptable.

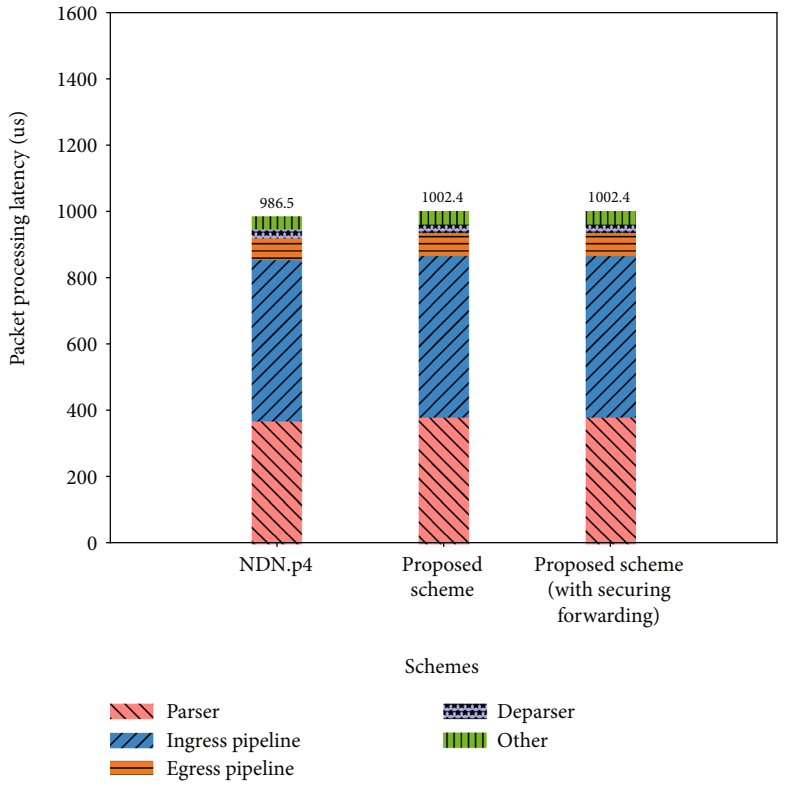
6.2.3. Packet Processing Latency. On the basis of content retrieval latency, we evaluate the packet processing latency between in different schemes, as shown in Figure 9.

Figures 9(a) and 9(b) show the comparison for main block processing latency of interest packets. The parser and ingress pipeline blocks account for the vast majority of the packet processing latency. Compared with the traditional NDN.p4 solution, the proposed scheme slightly increases the average packet processing latency. This is because the proposed scheme adds extra operations in the parser and pipeline in order to support the function of in-network caching and the retransmission of interest packets. In addition, enabling secure forwarding has no significant effect on the packet processing latency of interest packets. This is because the object of our encryption and decryption operation is the payload of the data packets, so it will not affect the packet processing latency of the interest packets.

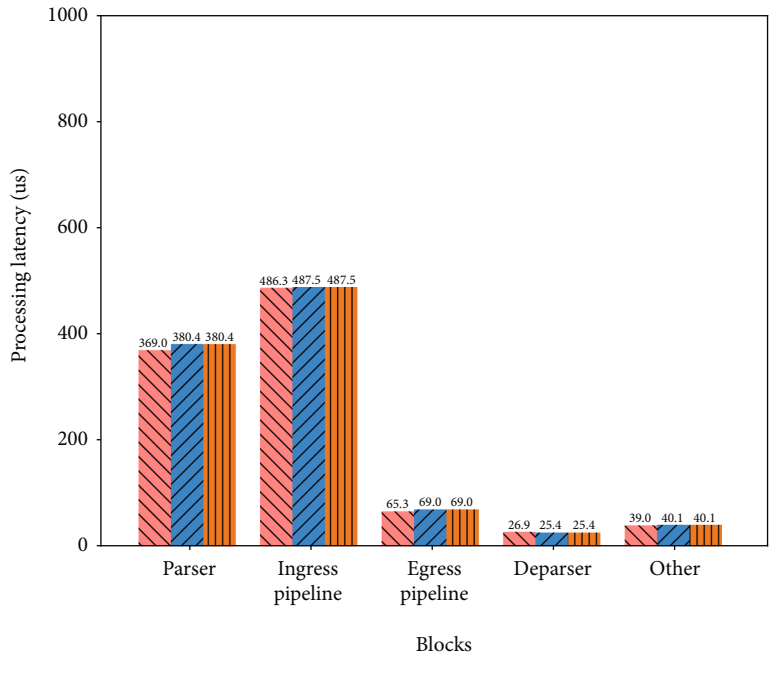
Table 2: The comparison of content retrieval latency.

Different schemes	From content producer	From cache
NDN.p4 [3]	6.350 ms	—
Proposed scheme	6.370 ms	2.727 ms
Proposed scheme (with secure forwarding)	7.172 ms	2.729 ms

Figures 9(c) and 9(d) show the comparison for main block processing latency of data packets. Compared with the traditional NDN.p4 solution, the proposed scheme slightly reduces the average processing latency in ingress and egress pipelines. This is thanks to our optimization of the traditional forwarding pipeline. In addition, when the secure forwarding function is enabled, the average packet processing latency increases by about 24.5%, and most of the added extra latency is in the ingress pipeline. This is because programmable switches need to perform additional

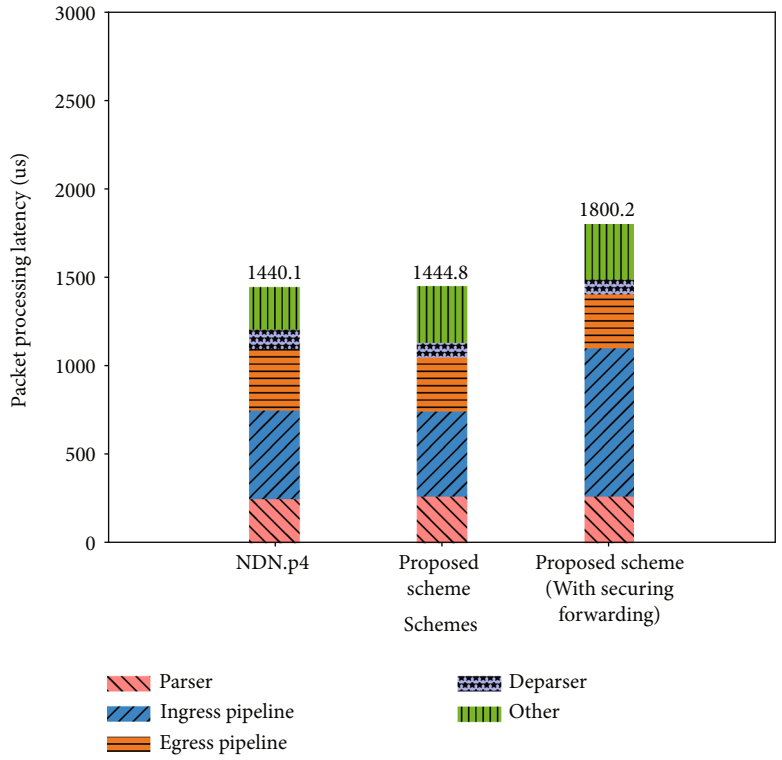


(a) Packet processing latency of interest packets

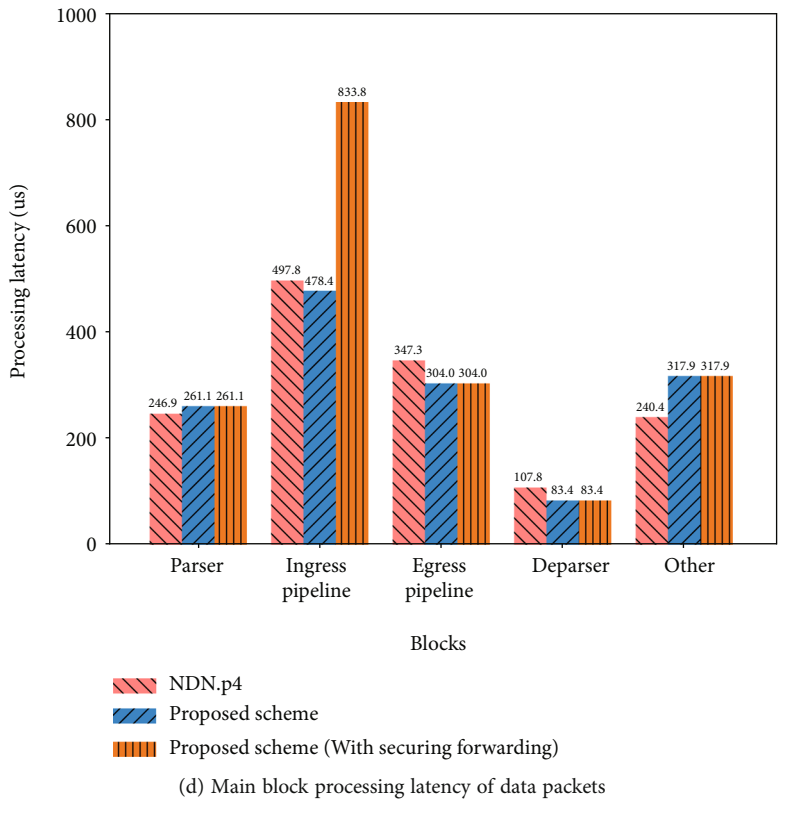


(b) Main block processing latency of interest packets

Figure 9: Continued.

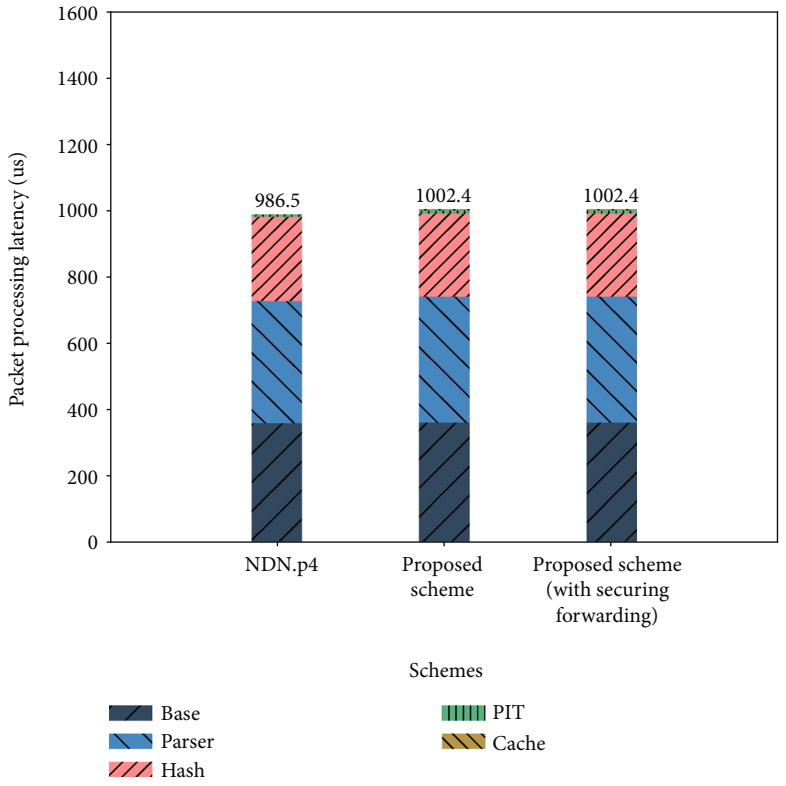


(c) Packet processing latency of data packets

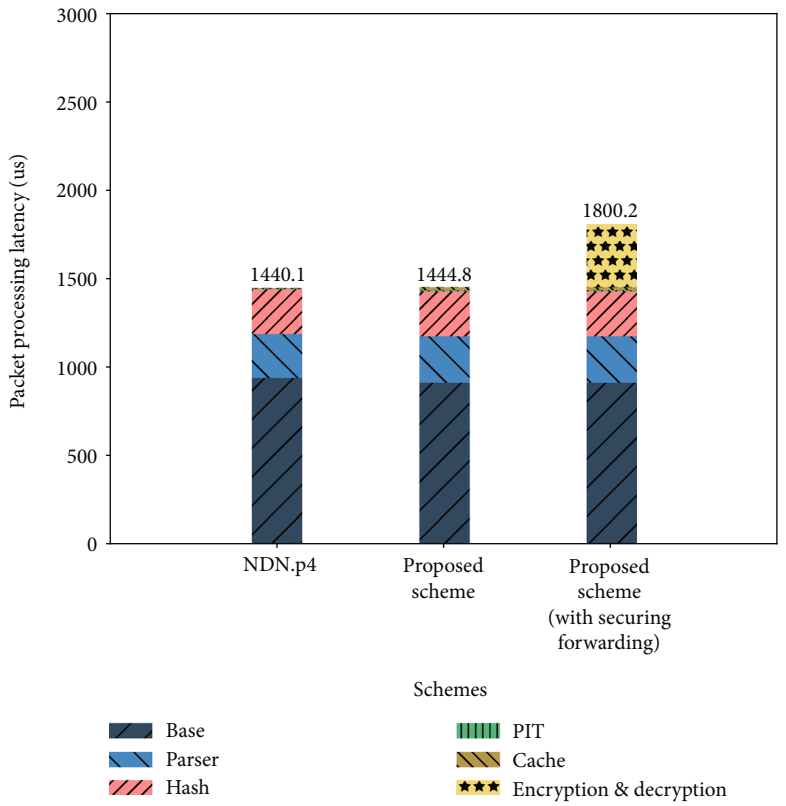


(d) Main block processing latency of data packets

Figure 9: Continued.



(e) Main operation processing latency of interest packets



(f) Main operation processing latency of data packets

Figure 9: The comparison for processing latency of P4 target.

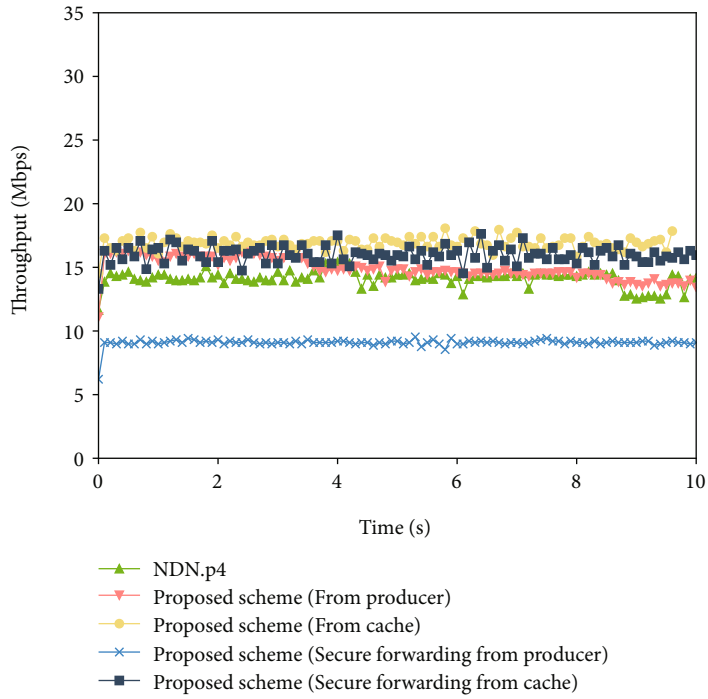


Figure 10: Comparison for received throughput.

operations in the ingress pipeline to complete the encryption and decryption of data packets.

Figures 9(e) and 9(f) show the main operation processing latency of interest/data packets in different schemes. The additional packet processing latency caused by enabling the in-network caching function is less than 0.5% of the total packet processing latency. This is due to the separation of the CS-list and the CS-server architecture. The in-network caching function is no longer a restriction on forwarding performance. In addition, when the secure forwarding function is enabled, an additional overhead of about 355 μs is added compared to the case where the secure forwarding function is not enabled, which is generated when the content permutation algorithm performs operations on the payload of the data packets. Since it only accounts for 20% of the total packet processing latency, the additional time cost is acceptable.

6.2.4. Received Throughput. Finally, we evaluate the received throughput of the proposed scheme. In our environment, the consumer requests a traffic of about 16 MB files. Furthermore, the proposed mechanism deploys a slow-start algorithm and a congestion avoidance algorithm for congestion control. We evaluate the receive throughput in different cases separately.

As shown in Figure 10, in the case of only enabling the cache function, the proposed scheme outperforms in the receive throughput compared with the traditional NDN.p4 solution, which is due to the fact that the proposed scheme supports the retransmission of interest packets and the multicast forwarding of data packets. In addition, when the secure forwarding function is enabled, the average received throughput drops to about 9 Mbps (from producer). It is worth noting that the traffic fluctuations at the end of the curve are related to retransmitted packets over a period of time, which is mainly due to the multithreaded design of BMv2. Every time a new data packet is cloned, a new thread will be created, which may lead to continuous processing of multicast forwarding. Multiple clone queues in a process cause high queuing delays for individual clone packets.

7. Security Analysis and Discussion

In this section, we describe the two main attacks and threats in the packet forwarding process. Next, we analyze how the proposed mechanism responds to these two attacks.

7.1. Man-in-the-Middle Attack. The man-in-the-middle attack is to hijack interest packets or the response data packets sent by the consumer through the intermediate switch and tamper with the content, to achieve the purpose of making the consumer unable to obtain the required information normally.

In the proposed scheme, the control plane periodically sends control signaling to the programmable switches located in the data plane through a TLS-encrypted control channel. The switch completes the pipeline operation (including encryption and decryption operations) of the data packets through control signaling. Even if a malicious device is connected to the network, since it does not have the control signaling periodically issued by the controller, it cannot complete the pipeline operation of the data packet to obtain the information of the data packet. Therefore, man-in-the-middle attacks can be mitigated.

7.2. Eavesdropping Attacks. Eavesdropping attacks mean that devices maliciously accessing the network can obtain rich content information in data packets forwarded in the network.

In the proposed scheme, we combine the content replacement algorithm to encrypt data packets with security requirements in the network. Since an attacker cannot brute force a key with a length of 128 bits or more in a short period of time, he cannot obtain the original content information, thereby mitigating the threat of eavesdropping attacks.

8. Conclusion

In this paper, we present the design of a secure and cached-enabled NDN forwarding plane based on programmable switches. To mitigate the impact on the forwarding plane performance, our scheme adopts a separate architecture of cache query and cache response. With the advantage of network programmability of P4 technology, we extend the content permutation algorithm and integrate it into NDN forwarding plane, which makes our scheme also support lightweight secure forwarding. In addition, we enhance the design of traditional NDN forwarding plane to support interest retransmission and multicast forwarding of data packets. Experiment results show that our scheme outperforms in terms of content retrieval latency and received throughput. Besides, our scheme can support on-demand secure forwarding with low cost.

Compared with software BMv2 switches, physical programmable switches have more powerful packet processing capabilities. Next, we will evaluate our proposed scheme on physical programmable switches.

Data Availability

All data generated or used during the study appear in the submitted paper.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (No. 2019YFB1802503) and the National Natural Science Foundation of China (Nos. 61972026 and 61802014).

References

[1] T. Barnett, S. Jain, U. Andra, and T. Khurana, "Cisco visual networking index (VNI) complete forecast update, 2017–2022," *APJC Cisco Knowledge Network (CKN) Presentation*, 2018, Available online: https://www.cisco.com/c/dam/m/en_us/network-intelligence/service-provider/digital-transformation/knowledge-network-webinars/pdfs/1213-business-services-ckn.pdf. Accessed 12 October 2022.

[2] B. Nour, S. Mastorakis, R. Ullah, and N. Stergiou, "Information-centric networking in wireless environments: security risks and challenges," *IEEE Wireless Communications*, vol. 28, no. 2, pp. 121–127, 2021.

[3] S. Signorello, R. State, J. Franois, and O. Festor, "NDN.p4: programming information-centric data-planes," *2016 IEEE NetSoft Conference and Workshops (NetSoft)*, pp. 384–389, IEEE Institute of Electrical and Electronics Engineers, Seoul, Korea, 2016.

[4] S. Signorello, *A Multifold Approach to Address the Security Issues of Stateful Forwarding Mechanisms in Information-Centric Networks*, University of Lorraine, Nancy, France, 2018.

[5] X. Guo, N. Liu, X. Hou, S. Gao, and H. Zhou, "An efficient NDN routing mechanism design in P4 environment," in *2021 2nd Information Communication Technologies Conference (ICTC)*, pp. 28–33, IEEE, Nanjing, China, 2021.

[6] S. Hou, Y. Hu, L. Tian, and Z. Dang, "NNFD.P4: NDN in-networking cache implementation scheme with P4," *IEICE Transactions on Information and Systems*, vol. E105.D, no. 4, pp. 820–823, 2022.

[7] X. Jin, X. Li, H. Zhang et al., "NetCache: balancing key-value stores with fast in-network caching," in *Proceedings of the 26th Symposium on Operating Systems Principles*, pp. 121–136, Association for Computing Machinery, New York, NY, United States, 2017.

[8] T. Qu, R. Joshi, M. C. Chan, B. Leong, D. Guo, and Z. Liu, "SQR: in-network packet loss recovery from link failures for highly reliable datacenter networks," in *2019 IEEE 27th International Conference on Network Protocols (ICNP). IEEE*, pp. 1–12, IEEE, Chicago, IL, USA, 2019.

[9] R. Miguel, S. Signorello, and F. M. V. Ramos, "Named data networking with programmable switches," in *2018 IEEE 26th International Conference on Network Protocols (ICNP)*, pp. 400–405, IEEE, Cambridge, UK, 2018.

[10] O. Karrakchou, N. Samaan, and A. Karmouch, "ENDN: an enhanced NDN architecture with a p4-programmabie data plane," in *Proceedings of the 7th ACM Conference on Information-Centric Networking, ser. ICN '20. New York, NY, USA: Association for Computing Machinery*, Association for Computing Machinery, p. 111, 2020, [Online]. Available.

[11] R. Tourani, S. Misra, T. Mick, and G. Panwar, "Security, privacy, and access control in information-centric networking: a survey," *IEEE Communications Surveys Tutorials*, vol. 20, no. 1, pp. 566–600, 2018.

[12] K. Xue, P. He, X. Zhang et al., "A secure, efficient, and accountable edge-based access control framework for information centric networks," *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, pp. 1220–1233, 2019.

[13] Z. Zhang, Y. Yu, S. K. Ramani, A. Afanasyev, and L. Zhang, "NAC: automating access control via named data," in *MILCOM 2018 – 2018 IEEE Military Communications Conference (MILCOM)*, pp. 626–633, IEEE, Los Angeles, CA, USA, 2018.

[14] S. Misra, R. Tourani, F. Natividad, T. Mick, N. E. Majd, and H. Huang, "AccConF: an access control framework for leveraging in-network cached data in the ICN-enabled wireless edge," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 1, pp. 5–17, 2019.

[15] L. Ning-Chun, G. Shuai, H. Xin-Di, and G. Xin-Chang, "A data access control scheme in information-centric mobile ad hoc networks," *Journal of Beijing University of Posts and Telecommunications*, vol. 44, no. 2, pp. 54–60, 2021.

[16] I. Oliveira, E. Neto, R. Immich et al., "Dh-aes-p4: on-premise encryption and in-band key-exchange in p4 fully programmable data planes," in *2021 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, pp. 148–153, IEEE, Heraklion, Greece, 2021.

[17] G. Liu, W. Quan, N. Cheng et al., "Softwarized IoT network immunity against eavesdropping with programmable data planes," *IEEE Internet of Things Journal*, vol. 8, no. 8, pp. 6578–6590, 2021.

[18] T.-T. Lin, S.-C. Tsai, and W.-G. Tzeng, "Efficient encoding and decoding with permutation arrays," in *2008 IEEE International Symposium on Information Theory*, pp. 211–214, IEEE, Toronto, ON, Canada, 2008.

[19] Y.-B. Lin, T.-J. Huang, and S.-C. Tsai, "Enhancing 5g/IoT transport security through content permutation," *IEEE Access*, vol. 7, pp. 94 293–94 299, 2019.

[20] A. Afanasyev, J. Shi, B. Zhang et al., "NFD developers guide," *Dept. Comput. Sci., Univ. California, Los Angeles, Los Angeles, CA, USA, Tech. Rep. NDN-0021*, 2014, https://named-data.net/publications/techreports/ndn-0021-11-nfd-guide/.

[21] P4lang/behavioral-model, 2022, https://github.com/p4lang/behavioral-model.