Per-Packet Protection (PPP) Scheme for Named Data Networking

Chen He[1,2], Jiangtao Luo[2], Fei Zhang[1,2], Zuoqi Jiang[1,2], Mengnan Wang[1,2]
1.School of Communications and Information Engineering, Chongqing University of Posts and Telecommunications
2.Electronic Information and Networking Research Institute, Chongqing University of Posts and Telecommunications
Chongqing 400065, China
erichec@outlook.com; Luojt@cqupt.edu.cn; S160131205@stu.cqupt.edu.cn; {294621293, 244791399}@qq.com

***Abstract*— Named Data Networking (NDN) is regarded as a promising architecture for the future Internet. Due to the characteristics of in-network caching and name-based routing in NDN, access control cannot be tied to a particular location, and traditional channel-based access control mechanisms are no longer viable, which brings a major challenge to the access control enforcement. To enhance content-based access control in NDN, this paper presents a per-packet protection (PPP) scheme based on a combination of public key encryption and symmetric-key cryptography, which adopts one-way hash functions to generate random cipher keys for different data packets. Furthermore, PPP using secret sharing method provides efficient and flexible access control, which supports scalability and collusion resistance. The experimental results prove that our solution introduces acceptable overheads and reduces the computation time at the users.**

***Index Terms*—Named Data Networking, access control, encryption, data confidentiality**

I. Introduction

With the popularity of Internet applications, Internet video streaming and downloads make 57% of all Internet traffic today and are expected to rise to 82% by 2021 [1], which implies that the multimedia content will constitute most of the Internet traffic and be shared. Meanwhile, with the explosive growth of Internet users, the TCP/IP-based network architecture displays deficiency in some aspects such as security, mobility and energy efficiency. As a set of clean-slate designs trying to address these issues, Information-Centric Networking (ICN) has been proposed, and NDN is one of the typical architectures. NDN retains the same layered hourglass architecture with functional differences, and adopts the forwarding mechanism based on the content name rather than the source and destination addresses in IP networks. Moreover, it leverages the in-network caching mechanism to reduce the network traffic load from redundant requests and improve the utilization of network resources [2].

Although NDN improves the network performance and keeps in line with these future traffic trends, it also bears a lot of security challenges, one of which is how to enforce the access control mechanism, in other words, how to only allow legitimate users to access valuable or even sensitive contents [3]. Due to the feature of in-network caching, NDN intermediate routers can cache the forwarded content, and future requests can be satisfied quickly from the cached duplicates, so the Content Producers (CPs) will lose direct control over their content, which implies that access control must be built into content itself [4].

Some encryption-based approaches have been applied to access control in NDN. Misra et al [5] [6] proposed an access control solution based on Broadcast Encryption (BE) where a symmetric key is distributed by BE. Tao Chen et al [7] proposed a probabilistic access control solution where a symmetric key is distributed using public key infrastructure (PKI) and NDN routers filter out unauthorized Interest packets via bloom filter. In general cases, this kind of access control is enough. However, if the same key is used to encrypt contents, an attacker could crack the key and decrypt the subsequent contents which will be fetched directly from NDN routers. So more secure access control for NDN is needed especially in some critical cases, such as the details of financial transactions, supply chain information in some companies. Besides, how to realize flexible and efficient revocation remains one of the major challenges for access control in NDN.

To address the issues, we propose per-packet protection (PPP) scheme that is based on the combination of public key cryptography, secret sharing method [8] and symmetric encryption. The main contributions of this paper are summarized as follows:

- Aiming at encrypting each Data packet with different keys, the symmetric key is generated using one-way hash functions which are lightweight enough, and easy to compute on every input, but hard to invert [9] [10], which avoids that authorized users frequently request keys for different packets.
- To distribute the symmetric key efficiently, our scheme combines Identity Based Cryptography (IBC) and an improved Shamir's Secret Sharing (SSS) Method. It is flexible in case of users' addition and deletion and reduces computation cost. Meanwhile, compared with the scheme in [5] [6], our solution using secret sharing can prevent users from colluding with better scalability.

The remainder of this paper is organized as follows. Section II provides the background about NDN, and some methods exploited in our scheme. Section III is devoted to the description

978-1-5386-4870-4/18/$31.00 ©2018 IEEE

of the proposed PPP scheme. Section IV and section V provide security analysis and performance evaluation respectively. Section VI concludes this paper.

II. Background

A. Named Data Networking

A communication session in NDN is initiated by a requester, and is performed using two packet types: Interest packet and Data packet. The requester puts the name of desired content into Interest packet and broadcasts it into the network. The router forwards Interest packet to the Content Producer (CP) based on that name. Once Interest packet arrives at a router caching the requested data, Data packet will be sent as a response from this router and follow the reverse path back to the requester.

Each router in NDN maintains the following three data structures, including Forwarding Information Base (FIB), Content Store (CS) and Pending Interest Table (PIT). FIB is similar to the routing table in IP networks, maintaining next-hop(s) and other information for each reachable destination name prefix. CS based on the cache strategy can selectively save Data packet for a period of time. PIT maintains an entry for each incoming Interest packet that the router has forwarded but not satisfied until its corresponding Data packet arrives or the entry lifetime expires. Each entry in PIT records the name and incoming interface of Interest packet so that Data packet can be forwarded downstream to the requester [11].

B. Shamir's Secret Sharing Method

Secret sharing method can be used to distribute the key and realize flexible access control. The method divides a secret into a number of shares, and no single share can independently reveal any information about the secret. The secret can be recovered only when the number of shares is greater than or equal to a specified value.

In Shamir's (k, n)-threshold scheme [8] based on polynomial interpolation ("Lagrange interpolation"), n shares will be generated from a secret, and at least k different shares are required to reconstruct the secret, but every group of less than k shares cannot obtain any information about the secret. The method generates a randomly chosen polynomial of (k-1)-degree: $f(x) = s + a_1x + a_2x^2 + \cdots + a_{k-1}x^{k-1}$, where s represents the secret and can be obtained by k different points (x_i, y_i) using Lagrange interpolation [12] as:

$$s = \sum_{i=1}^{k} y_i \times \prod_{j=1, j\neq i}^{k} \frac{-x_j}{x_i - x_j}, \tag{1}$$

where $y_i = f(x_i)$ represents the i-th share ($i = 1, 2, \ldots, n$).

C. Identity Based Cryptography

Unlike a traditional PKI system, an arbitrary string, such as a user's name, address or their combinations, can be chosen as a public key and represents a unique identity in IBC which includes Identity Based Encryption (IBE) and Identity Based Signature (IBS). The corresponding secret key is generated by a private key generator (PKG) and a semi-trusted third party. Thus, neither does any pair of users need to exchange private or public keys to communicate securely, nor is certificate authority (CA) indispensable to keep key directories.

The IBE scheme consists of the following four steps [13]:

- **Setup**: The PKG generates a master-secret key (MSK) and system parameters (SP) when inputting a security parameter k. Only SP are made publicly available.
- **Extract**: The PKG generates a secret key SK_{ID} when inputting SP, MSK, and an identity ID.
- **Encryption**: Take as input an identity ID, a message M, and SP, and as output a ciphertext $M^{'}$.
- **Decryption**: Take as input $M^{'}$ and the corresponding SK_{ID}, and as output M.

The IBS scheme consists of the following four steps [14]:

- **Setup** & **Extract**: The two steps are the same as the first two steps in IBE.
- **Encryption**: Take as input SP, M, and SK_{ID}, and as output the signature information σ that will be sent to the recipient with M together.
- **Decryption**: Take as input SP, M, the corresponding ID, and σ, and as output V. Only if V is equal to 1, the signature is valid.

D. One-way hash functions

In order to achieve different keys for different Data packets efficiently and ensure higher security, one-way hash functions that map data of arbitrary length to data of a fixed length are introduced to generate Data key. The one-way hash function H includes the following properties [9]:

- H can be applied to a block of data at any size.
- H produces a fixed length output.
- $H(x)$ is easy to compute for any given x.
- For any given block x, it is computationally infeasible to find $x \neq y$ with $H(x) = H(y)$.
- It is computationally infeasible to find any pair (x, y) such that $H(x) = H(y)$.

III. Proposed Solution

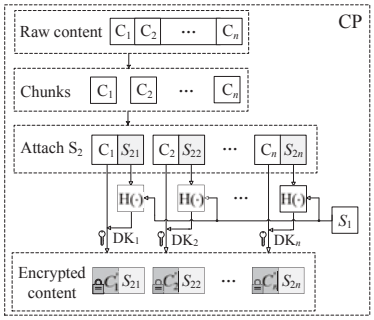
A. Per Packet Encryption

The proposed scheme is based on content encryption to protect the valuable or sensitive data objects. It encrypts each data packet with different cipher key. Each key consists of two parts, a fixed part called S_1 and a variable part called S_2. S_1 is invariant for all packets belonging to the same content while S_2 is different for different packets.

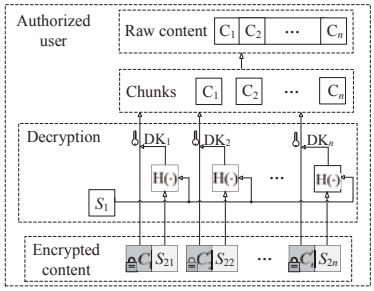
As shown in Fig. 1(a), a CP produces S_1 for each content, and divides the raw content into n smaller chunks $C_i, i = 1, \ldots n$; then generates n random strings for each of them as S_2 $(S_{21}, S_{22}, \ldots, S_{2n})$. The CP then generates different Data keys (DK_i) for each chunk using multilevel hash of S_1 and S_{2i} , $H(S_1, S_{2i})$, and then encrypts these chunks into ciphered messages $C_i^{'}$, taking DK_i as the symmetric cipher key. S_{2i} is appended to the message, building the ciphered Data packets. S_1 is distributed to the network by a secret shared method

described later, and only authorized users can restore it. $H(\cdot)$ represents the one-way hash functions.

On the other end, after requesting and obtaining the ciphered Data packets, an authorized user obtaining S_1 and knowing the Hash functions $H(\cdot)$ beforehand, can re-calculate DK_i using $H(S_1, S_{2i})$, then decrypt and get each chunk, and finally rebuild the original content. The procedure is illustrated in Fig. 1(b), .



(a) PPP on the CP



(b) PPP on the user

Fig. 1. The principle of PPP

B. Distribution of S_1

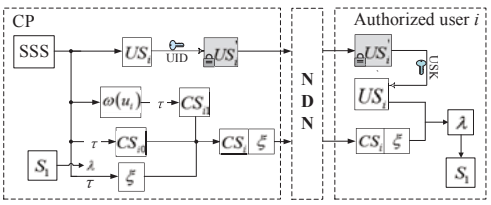


Fig. 2. The distribution of S_1

It is crucial for the proposed scheme to distribute S_1 securely and effectively. As described above, Shamir's secret sharing method (SSS) provides an effective way to distribute a secret among users, but Shamir's native approach is short in scalability since it links the polynomial degree to the number of users. Before introducing our modified version, we first define two basic concepts, which are enlightened by the idea in [15]:

Algorithm 1 Generation of Shares

Input: P, Q, k<Q, m<Q, where P, Q are large prime numbers, such that $P = rQ + 1$, where r is a positive integer (e.g. r = 2), and k, m represent the polynomial degree and the number of users respectively.

Output: Generates a polynomial $q(x)$ and $(x_j, q(x_j))$, $(u_i, q(u_i))$, US_i, $\omega(u_i)$, CS_i.

1: Generates $s, a_1, a_2, \cdots, a_k$ in the field Z_Q^*, and obtains $q(x) = s + a_1x + a_2x^2 + \cdots + a_kx^k$, where Z_Q^* is the multiplicative group of integers of order Q.

2: Generates and stores $(u_i, q(u_i))$, $(x_j, q(x_j))$, and ensures that u_i, x_j in the field Z_Q^* are positive, unique, where $i = (1, 2, \cdots, m), j = (1, 2, \cdots, k)$.

3: Calculates $US_i = \prod_{j=1}^{k} \frac{-x_j}{u_i - x_j}$.

4: Calculates and stores $\omega(u_i) = \sum_{j=1}^{k} q(x_j) \frac{-u_i}{x_j - u_i} \prod_{l=1, l\neq j}^{k} \frac{-x_l}{x_j - x_l}$, where g is a generator of a subgroup of G_P of order Q, and G_P is a cyclic group of order P.

5: Generates τ in the field Z_Q^*, and obtains $\xi_\eta = \lambda_\eta \cdot g^{\tau \cdot s}$, then calculates CS_i including $CS_{i0} = g^{\tau \cdot q(u_i)}$, $CS_{i1} = g^{\tau \cdot \omega(u_i)}$, where $\eta = (1, 2, \cdots, \mu)$, and λ_η is the S_1 or its sub-strings in the field Z_Q^*.

User's share (US): It is a number computed by the CP for each authorized user, and will be securely sent to the user. It is a user-dependent parameter for re-constructing S_1.

Complementary share (CS): It is another number calculated by the CP for each authorized user when each content is produced. It is a user-dependent and content-related parameter for re-constructing S_1.

Given US and CS, a user can reconstruct S_1. Algorithm 1 presents the procedure for generation of these shares. For a bigger and more complex S_1, the CP will split it into smaller sub-strings λ: $S_1 = \lambda_1 \| \lambda_2 \| \cdots \| \lambda_\mu$. The CP picks a random k-degree polynomial $q(x) = s + a_1x + \ldots + a_kx^k$, and generates k points in a 2-dimensional plane $(x_1, q(x_1)), (x_2, q(x_2)), \ldots, (x_k, q(x_k))$ with distinct x_j ($j = 1, 2, \ldots, k$).

For a certain user i, a random point $(u_i, q(u_i))$ is selected, such that US_i for this user is calculated by $US_i = \prod_{j=1}^{k} \frac{-x_j}{u_i - x_j}$. To calculate CS_i of the user, a parameter $\omega(u_i) = \sum_{j=1}^{k} q(x_j) \frac{-u_i}{x_j - u_i} \prod_{l=1, l\neq j}^{k} \frac{-x_l}{x_j - x_l}$ is calculated. Given g, a generator of a subgroup of G_P of order Q, and τ generated for a content object, the CP can obtain $\xi_\eta = \lambda_\eta \cdot g^{\tau \cdot s}$, $\eta = (1, 2, \cdots, \mu)$, and then get the CS_i containing $CS_{i0} = g^{\tau \cdot q(u_i)}$ and $CS_{i1} = g^{\tau \cdot \omega(u_i)}$ for the user. Note that τ varies for different content objects, leading to various CS_i for the same user i. Fortunately, the CP can compute different CS_i rapidly for different content objects because $\omega(u_i)$ is stored in CP.

In Shamir's native approach, a point $(u_i, q(u_i))$ is assigned

to the corresponding user. However, in our scheme, the users hold *US* instead of the coordinates, and they cannot use the other users' *US* to reconstruct S_1. Consequently, the *US* and *CS* introduced can make the polynomial degree no longer related with the number of users, which avoids the collusion problem and provides sufficient scalability. Meanwhile, once a user fetches *US*, it can request *CS* for different contents, and the CP will assign the corresponding *CS* to the user directly without encryption, which can make the distribution of S_1 more efficient.

Considering the major strength of IBC scheme which can establish a secure channel over a non-secure medium, the *US* is distributed using IBC scheme.

A trusted PKG for IBC is responsible for generating the secret key and common system parameters in the network. Meanwhile, each user or producer has its own identity and the corresponding secret key, namely the user's secret key (USK) and the user's identity (UID), the producer's secret key (PSK) and the producer's identity (PID). Each entity uses UID or PID as the public key in the scheme.

As shown in Fig. 2, the CP calculates and encrypts US_i using UID, then distributes it to the authorized user i, and the user can obtain US_i using USK. In addition, the CP generates S_1 for a content object, then calculates and distributes ξ, CS_i, and the user i calculates $S_1 = \lambda_1 \| \lambda_2 \| \cdots \| \lambda_\mu$ based ξ, CS_i and US_i:

$$\begin{aligned} \lambda_\eta &= \frac{\xi_\eta}{CS_{i1} \cdot CS_{i0}^{US_i}} \\ &= \frac{\xi_\eta}{g^{\tau \cdot \omega(u_i)} g^{\tau \cdot q(u_i) \cdot US_i}} \\ &= \frac{\lambda_\eta \cdot g^{\tau \cdot s}}{g^{\tau \cdot s}}, \eta \in [1, \mu]. \end{aligned} \tag{2}$$

Fig. 3. The modified structure of Interest packet

C. Extension of Interest Packet

In order to verify whether the requester is legitimate, Interest packet will be extended when using PPP scheme. When requesting the legitimate access right or *CS*, a user encrypts its UID using the CP's PID, and embeds the encrypted UID within the Interest packet as Identity, which can effectively protect the user's identity and individual privacy. Meanwhile, Timestamp that is unique for the same user in Interest Packet can prevent the malicious eavesdropper from using the legitimate requests. If an Interest packet with the duplicate Timestamp and Identity is transmitted to a CP, it can be regarded as a bogus request and dropped. In addition, Signature is generated using USK to verify the authenticity of the user further. The modified structure of Interest packet is illustrated in Fig. 3.

D. Procedure of PPP

The procedure of PPP scheme includes the following steps:

Initialization. The CP generates the polynomial of k-degree and obtains the shares.

Registration. To obtain access right, a user must firstly send an Interest packet for registration. The UID's hash value is added to the request name, which protects the users' privacy and differentiates different users' requests. The user gets $Identity = \textbf{IBE}.encrypt(UID, PID)$, where $\textbf{IBE}.encrypt(\cdot)$ is the encrypting algorithm of IBE scheme, then has $Signature = \textbf{IBS}.sign((Name, Timestamp), USK)$, where $\textbf{IBS}.sign(\cdot)$ is the signing algorithm of IBS scheme. The $Identity$, $Timestamp$ and $Signature$ are to be transmitted together with Interest packet.

Distribution of *US*. After receiving the request, the CP verifies the validity of the request by $Identity$ and $Timestamp$ in the Interest packet. The CP firstly uses its own PSK to decrypt the $Identity$, and obtains $UID = \textbf{IBE}.decrypt(Identity, PSK)$, where $\textbf{IBE}.decrypt(\cdot)$ represents the decrypting algorithm of IBE scheme. To determine if the request is bogus and eliminate the asynchronism between the clocks of CP and user, the CP must maintain a collection of timestamps for the same user in a certain period of time T. Once a duplicate $Timestamp$ for the same user is accepted within T, the CP will drop the packet. Then in order to verify the authenticity of the user further by checking the $Signature$, the CP obtains $Result = \textbf{IBS}.verify((Name, Timestamp), Signature, UID)$, where $\textbf{IBS}.verify(\cdot)$ represents the verifying algorithm of IBS scheme. Only if $Result$ is TRUE, the CP will generate US_i for the user, then use UID to encrypt US_i, and obtain $US_i^{'} = \textbf{IBE}.encrypt(UID, US_i)$. The Data packet containing $US_i^{'}$ will be returned to the user. When receiving the Data packet, the user obtains $US_i = \textbf{IBE}.decrypt(US_i^{'}, USK)$.

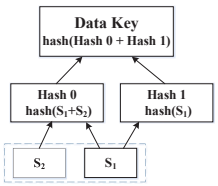


Fig. 4. The generation of DK

DK generation and Content encryption. The method $H(S_1, S_2)$ generating DK is similar to the Merkle Hash Tree (MHT) algorithm [16], as shown in Fig. 4. Firstly, the CP generates S_1 and random S_2, then obtains DK by $H(S_1, S_2)$. DK may be the top hash value or only portions of the top hash value. Finally, the CP performs the symmetric encryption

using DK and distributes the ciphered message. Hash functions mainly used to generate DK make all operations lightweight.

Content decryption. To access one content object, the authorized user firstly initiates an Interet packet requesting the responding CS, and calculates S_1, then extracts S_2 and calculates DK by $H(S_1, S_2)$. Finally, the user retrieves the original message by performing the symmetric decryption.

User revocation and addition. The addition and revocation of access privileges in PPP scheme will not affect other legitimate users. When revoking a user u_x, the CP only calculates $CS_i, (i = 1, 2, \cdots, m, i \neq x)$. If the user u_x requests CS_x, the CP will return a prompt message without CS_x. In addition, when adding a new user, the CP will generate new $(u, q(u))$ which must be not reused for other users, then calculate *US* and $\omega(u)$ for it.

IV. Security Analysis

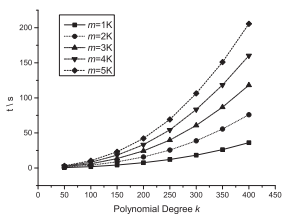
In this section, we verify the security of the PPP scheme. Due to the SSS adopted here, we firstly analyze the collusion resistance. Each legitimate user only possesses one incomplete and specified share called *US* here, and does not hold the point $(u, q(u))$, so *US* must be combined with the corresponding *CS* to recover S_1, and the polynomial degree is no longer associated with the number of users, which can make the collusion of n+1 users very hard.

Then we analyze the use of one-way hash functions which are very effective to generate key for each chunk. There are mainly two common methods attacking the hash funcitons. One is exhaustive method, it is easily implemented only when the password is simple. However, the method is difficult to work in the complicated situation, especially for the input with the variable length. S is the input in the one-way hash functions, the length of S is L, the number of different characters may appear in S is N, and the time required to crack the hash value each time is T_0. So the total time to crack is $T = T_0 \times N^L$. With the increments of N and L, the attacker needs more time to crack, so it is hard to obtain the original value by using the ordinary exhaustive method. Another is to find collision. When the collision happens, the hash values of different strings are same. But there is no effective way to find collision for SHA1, for example, the length of hash value for SHA1 is 160 bit, so there are 2^{160} results, which reveals that collision probability is quite low. In order to improve the security further, PPP scheme adopts the MHT algorithm with multiple hash functions. If S_1 is complex enough, the attacker cannot crack it by using the above methods. Even though S_1 is simple, the attacker cannot obtain the result within a reasonable time.

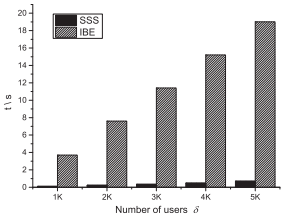
V. Performance Evaluation

In order to demonstrate the feasibility of the proposed scheme for NDN environment, we evaluate its efficiency, including the running time of generating shares and calculating S_1, computation cost and system resource consumption when encrypting and decrypting chunks at CP and at users respectively in the testbed based on NFD [17]. All simulation experiments are performed on the Ubuntu 16.04 with 2 GB RAM and a 4-core CPU. All the programs are written in the C++ language using the GNU Multiple Precision Arithmetic (GMP) [18] library and Number Theory Library (NTL) [19] for cryptographic operations.

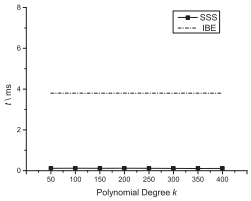
A. Efficiency of SSS



(a) Generation time of users and complementary shares



(b) Generation time of *CS*



(c) S_1 calculation time on the user

Fig. 5. The computation cost of SSS

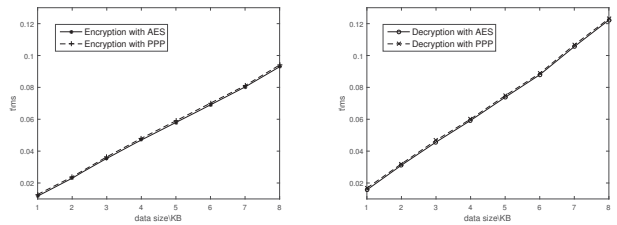
We implement the improved SSS, and measure the computation cost of generating the shares at the CP and recovering S_1 at the user. Note that k ranges from 50 to 400 in increments of 50, and m ranges from 1K to 5K in increments of 1K, where K stands for thousand. The results are averaged over 100 experiments run.

Fig. 5(a) shows the time consumed by the CP to generate a polynomial, users' and complementary shares. The running time is dependent on the value of k and m. As the numbers of k and m increase, the required time increase. Due to the computation of *CS* and *US*, the procedure consumes more time, but it can be performed offline by the CP. Additionally, the scalability and collusion issues have been solved in the improved SSS. Therefore, to reduce the computation cost at the CP, we can choose a moderate-sized k. Note that Fig. 5(a) displays the overall running time of Algorithm 1, however, the CP only calculates *CS* based $\omega(u)$ for a content data when there are no new users to be added. Fig. 5(b) shows the time taken to generate *CS* that noly performs the step in Line 5 of Algorithm 1. Compared with IBE, the improved SSS that generates and assigns *CS* to the corresponding user without IBE encryption will take less time on the CP. Fig. 5(c) presents a comparative analysis of the S_1 calculation time in SSS and IBE on the user. Due to the pre-computation of the Lagrangian coefficients, the CP does not transmit the actual point $(u, q(u))$ but *US* and *CS*, resulting in reduced computation time at the user. Compared with IBE, the PPP using SSS to distribute the

S_1 for different contents will realize a more efficient access control scheme on users' level.

B. Efficiency of Encryption/Decryption

In our experiments, Advanced Encryption Standard (AES) is chosen to encrypt data objects, and Secure Hash Algorithm 1 (SHA1) is chosen as the one-way hash function which calculates the DK here. Besides, we choose 128 bits as key size.



(a) Encryption with AES and PPP (b) Decryption with AES and PPP

Fig. 6. Speed of Encryption/Decryption with AES and PPP

First of all, eight text files treated as Data packets with different sizes ranging from 1KB to 8KB in increments of 1KB, are encrypted and decrypted using AES encryption algorithm with and without SHA-1. The AES without SHA-1 and with SHA-1 represents the traditional encryption-based access control mechanism and our PPP scheme respectively. As shown in Fig. 6, it is not surprising to see that the PPP scheme introduces a litte more overhead on encryption and decryption steps.

C. System Overhead

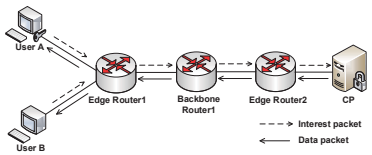
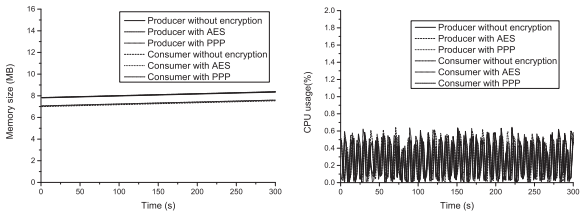


Fig. 7. Topology

TABLE I
THE KEYS OF PARTIAL DATA PACKETS

The chunk name(S_2)	S_1	**DK**
/content /text/001	a735146778e7 1b17e8f25580 6cf83223511b b2756cf842ef f6cb9b8a39f8 7098	565789b7cf23d60 67b1efc4835d668d
/content /text/002		aa2fc79ee835577 43281bf25ca75a9f7
/content /text/003		3c8a2fd4e112843 073e01ff481d47de6

Next, we compare the impact on the system usages, including the memory consumption and CPU usage, on the CP and user. We implement the PPP scheme based on NFD. For simplicity, the topology is a simple line with three nodes shown in Fig. 7 where the user A and user B represent the authorized and unauthorized users respectively. The users send an Interest packet per second, and the CP sends the text files with the size of 2K bytes as the protected Data packets. Note that the names of Data packets are chosen as S_2 here, since the chunks' names are different without increasing the additional message and communication cost.



(a) Memory usage (b) CPU usage

Fig. 8. The comparison of memory and CPU usage

The experimental results show that only the user A can decrypt the text segments while the user B cannot read the text segments easily in the PPP scheme even though it can fetch these text chunks from the intermediate NDN routers. The keys of partial text chunks are displayed in Table I. In Table I, S_1 and DK are printed out and displayed in hexadecimal format. Fig. 8 shows that the memory and CPU usage on the CP and user in our scheme nearly remain the same for the scenarios with or without encryption-based access control. In other words, the PPP scheme using one-way hash functions does not incur significant memory and CPU overhead on both the CP and the user.

VI. CONCLUSION

In this paper, we propose a PPP scheme which ensures that each protected Data packet will obtain higher security. The data objects are encrypted using different symmetric keys, and the fixed part of the keys is distributed to authorized users by IBC and SSS, which ensures that only authorized users can obtain the keys of content data using one-way hash functions and provides more efficient access control. Our experimental results have demonstrated the performance of the proposed solution, which reduces computation cost on users' level and takes less time to distribute each content object's key on the CP, and introduces the acceptable overheads of the content encryption and decryption process.

ACKNOWLEDGMENT

The work was jointly supported by the Chongqing Municipal project under GRANT cstc2015jcyjBX0009 and CSTCKJCXLJRC20.

REFERENCES

[1] C. V. networking Index, "Forecast and methodology, 2016-2021, white paper," *San Jose, CA, USA*, 2016.

[2] S. Wang, J. Bi, J. Wu, Z. Li, W. Zhang, and X. Yang, "Could in-network caching benefit information-centric networking?" in *Asian Internet Engineering Conference*, 2011, pp. 112–115.
[3] R. Tourani, S. Misra, T. Mick, and G. Panwar, "Security, privacy, and access control in information-centric networking: A survey," *IEEE Communications Surveys and Tutorials*, vol. PP, no. 99, pp. 1–1, 2016.
[4] X. Zhang, K. Chang, H. Xiong, Y. Wen, G. Shi, and G. Wang, "Towards name-based trust and security for content-centric network," in *IEEE International Conference on Network Protocols*, 2011, pp. 1–6.
[5] S. Misra, R. Tourani, and N. E. Majd, "Secure content delivery in information-centric networks: design, implementation, and analyses," in *ACM SIGCOMM Workshop on Information-Centric NETWORKING*, 2013, pp. 73–78.
[6] S. Misra, R. Tourani, F. Natividad, T. Mick, N. E. Majd, and H. Hong, "Accconf: An access control framework for leveraging in-network cached data in the icn-enabled wireless edge," *IEEE Transactions on Dependable & Secure Computing*, vol. PP, no. 99, pp. 1–1, 2016.
[7] T. Chen, K. Lei, and K. Xu, "An encryption and probability based access control model for named data networking," in *PERFORMANCE Computing and Communications Conference*, 2014, pp. 1–8.
[8] A. Shamir, *How to share a secret*. ACM, 1979.
[9] M. Naor and M. Yung, "Universal one-way hash functions and their cryptographic applications," 1989, pp. 33–43.
[10] Q. Li, X. Zhang, Q. Zheng, and R. Sandhu, "Live: Lightweight integrity verification and content access control for named data networking," *Information Forensics and Security IEEE Transactions on*, vol. 10, no. 2, pp. 308–320, 2015.
[11] D. Saxena, V. Raychoudhury, N. Suri, C. Becker, and J. Cao, "Named data networking: A survey," *Computer Science Review*, vol. 19, pp. 15–55, 2016.
[12] D. A. Quadling, *Lagrange's Interpolation Formula*. John Wiley & Sons, Inc., 2006.
[13] B. Dan and M. Franklin, "Identity-based encryption from the weil pairing," *Siam Journal on Computing*, vol. 32, no. 3, pp. 213–229, 2001.
[14] E. Kiltz, G. Neven, and M. Joye, "Identity-based signatures," *Cryptology & Information Security*, 2008.
[15] Y. Imine, A. Lounis, A. Bouabdallah, Y. Imine, A. Lounis, A. Bouabdallah, Y. Imine, A. Lounis, and A. Bouabdallah, "Abr: A new efficient attribute based revocation on access control system," in *International Wireless Communications and Mobile Computing Conference*, 2017, pp. 735–740.
[16] G. Becker, "Merkle signature schemes, merkle trees and their cryptanalysis," 2013.
[17] A. Afanasyev, J. Shi, B. Zhang, L. Zhang, I. Moiseenko, Y. Yu, W. Shang, Y. Huang, J. P. Abraham, S. DiBenedetto *et al.*, "Nfd developers guide," *Dept. Comput. Sci., Univ. California, Los Angeles, Los Angeles, CA, USA, Tech. Rep. NDN-0021*, 2014.
[18] "The gmp library," https://gmplib.org/.
[19] "Ntl: A library for doing number theory," http://www.shoup.net/ntl/.