A Novel Certificateless Signature-based Access Control Scheme for Named In-network Computing Service

Wanji Li, Qiangbin Liu, Yi Zhu

Abstract—**Named in-network computing service (NICS) is a potential computing paradigm emerged recently. Benefitted from the characteristics of named addressing and routing, NICS can be flexibly deployed on NDN router side and provide nearby computing service to Internet users. But the NICS feature of dynamic deployment also causes serious security risk of access control. How to independently check out the subscription relationship between requester and requested computing service on NICS side become a challenge. To solve this problem, we propose a novel certificateless signature-based access control scheme (CS-ACS) in this paper. In CS-ACS, the entire user public-private key pairs consist of two parts, user side and source server side. Where, the user public-private key pairs (server side) are generated according to the user ID, service subscription relationship and subscription expiration time. Based on this special design, when authorized user signs the interest packet of invoking specific service using its private key, the NICS can verify the signature then check out whether the requester is a valid subscriber and the subscription is expired or not. Simulation results show that, comparing with fundamental solutions, CS-ACS can avoid extra secret key storage cost on NICS side and markedly shorten authentication delay.**

Index Terms—**Named Data Networking, In-network Computing, Access Control, Subscription Relationship Verification**

I. Introduction

NAMED Data Networking (NDN)[1] is a famous candidate of next generation Internet architecture. Through introducing the named routing and in-network caching, NDN achieves effective distribution capability for static content. But, with the development of ubiquitous intelligent applications, invoking computing service from Internet has become a more urgent requirement of current Internet user than fetching static content[2][3]. Facing the fast-growing computing traffic pressure of network edge and cloud, recently, deploying lightweight Virtual Machine (VM) on NDN router side to provide named in-network computing service (NICS) is emerging as a potential solution[4][5].

Manuscript received March 9, 2024; revised November 21, 2024. This work was supported by in part by the National Natural Science Foundation of China (62276116), Future Network Scientific Research Fund Project of Jiangsu Province (FNSRFP-2021-YB-49) and the National College Student Innovation and Entrepreneurship Training Program Project (202410299052Z).

Wanji Li is a postgraduate student at School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, China (e-mail: 2677621559@qq.com).

Qiangbin Liu is an undergraduate student at School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, China (e-mail: 2992824314@qq.com).

Yi Zhu is a Professor of School of Computer Science and Communication Engineering, the Dean of Department of Communication Engineering, Jiangsu University, Zhenjiang, China (e-mail: zhuyi@ujs.edu.cn).

Figure.1 shows a demonstration of virtualization-based NICS, where the NDN router is a commercial server integrating with the virtual NDN routing function [6](e.g., a VM running the NDN Forwarding Daemon), each source server runs several computing services while providing the VM or container images of these services. According to the statistics of received requests and current free resources, each NDN router independently retrieves the required computing service VM image or container image from source server or nearby routers, then loads it as local computing service. When an interest packet of invoking computing service arrives, the router first checks whether the target service is running on it or not. If existing, the interest packet will be forwarded to corresponding computing service VM for further processing locally; otherwise, the interest packet will be forwarded to next router. Figure.2 further disclose the internal structure of the virtualization-based NDN router. To provide the routing function, one or more VMs are deployed on it to run the NDN Forwarding Daemon. The rest resources can be used to load the VMs of running the NICSs. The communications between VMs is based on the Open vSwitch.

Obviously, the virtualization-based NICS allows the computing service to be flexibly deployed and executed anywhere, the named addressing mechanism of NDN makes the service discovery and VM migration to be easily than TCP/IP architecture[7]. Benefitted from these characteristics, NICS can help Internet users to obtain required computing service from nearby routers instead of visiting Cloud servers, then markedly shorten the service invoking delay. But NICS also causes several new security risks[8][9], one important security issue is the access control which is also the typical security topic of NDN[10]. Access control aims at distinguishing the requester permission at router side, then forbidding the unauthorized requester to obtain the resources. For the scenario of static content distribution, access control means who can fetch or consume the content cached in the router. For the scenario of virtualization based NICS, the access control means who can invoke the computing service deployed in the router.

Figure 1 also shows the typical NICS invocation process, including three "interest-data" interaction stages[11], where the access control mechanism works on the first stage.

(1) Stage 1 (Initialization stage): When a NICS receives the interest packet of invoking its computing service, it will identify the invoker's identification first. If the interest packet comes from an authorized user, the NICS will reply with an acknowledgement data packet with estimated task complete time and result location name; Otherwise, the NICS will deny this request.

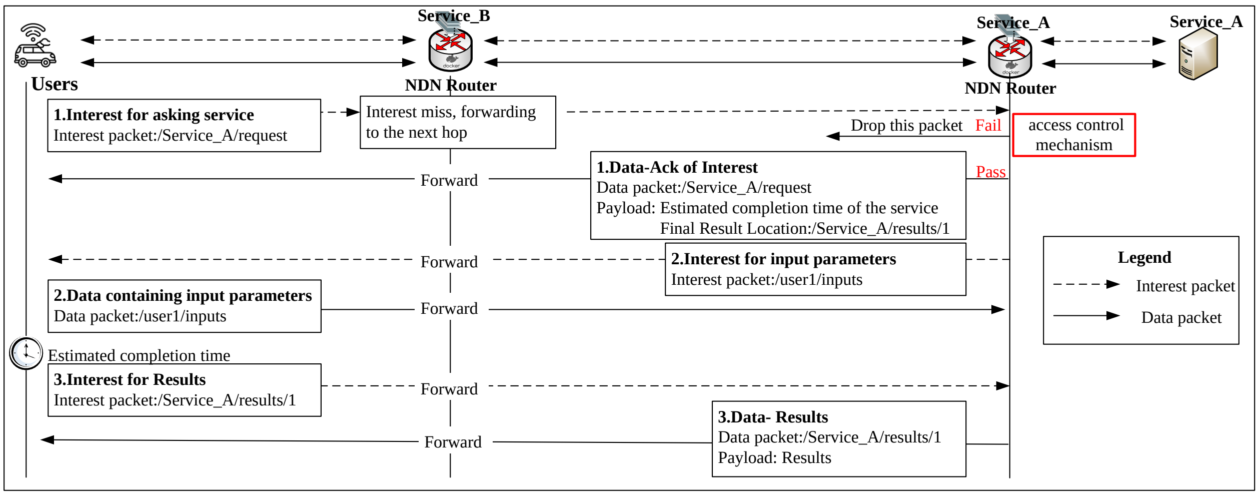
Users
Service_B
NDN Router
Service_A
NDN Router
Service_A
1.Interest for asking service
Interest packet:/Service_A/request
Interest miss, forwarding to the next hop
Drop this packet Fail
access control mechanism
Pass
Forward
1.Data-Ack of Interest
Data packet:/Service_A/request
Payload: Estimated completion time of the service
Final Result Location:/Service_A/results/1
2.Interest for input parameters
Interest packet:/user1/inputs
2.Data containing input parameters
Data packet:/user1/inputs
Estimated completion time
3.Interest for Results
Interest packet:/Service_A/results/1
3.Data- Results
Data packet:/Service_A/results/1
Payload: Results
Legend
Interest packet
Data packet

Fig. 1: Network Model of CS-ACS

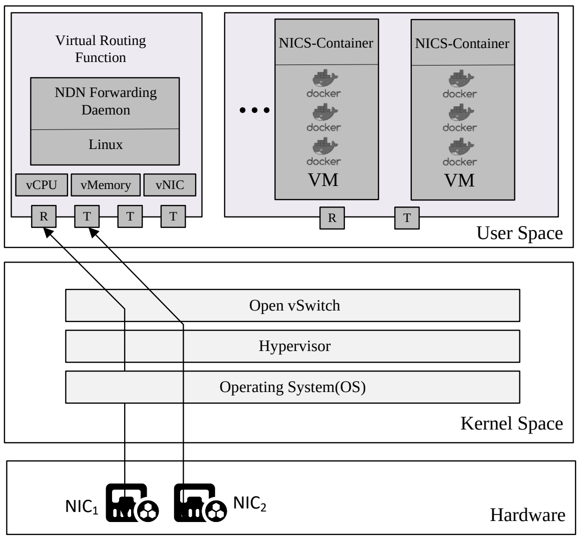

Fig. 2: The Internal Structure of the Virtualization-based NDN Router

(2) Stage 2 (Parameters retrieving stage): After identifying the invoker's permission, the NICS will actively send an interest packet to retrieve the computing input parameters from the authorized user, then the user replies with a data packet containing required data.

(3) Stage 3 (Computing result return stage): The user sets a timer according to the estimated complete time. Once the timer expires, the user will produce an interest packet using the result location name to fetch the final result. After obtaining the result, this computing service invocation is over.

From the above process, we can see the importance of access control. To prevent the unauthorized users from maliciously invoking computing service and occupying the router's resources, identifying the unauthorized users and denying their requests is necessary for NICS. How to design the access control scheme under NDN architecture, there are two available solutions, one is to encrypt the fetched/retuned data packet, another is to sign the requesting interest packet.

For the former solution, the fetched/retuned data packet is always encrypted by the requester's public key, then only the user owning the corresponding private key can consume this encrypted data. But this way isn't suitable for NICS scenario. Once the data packet is generated under NICS, it means the NICS has already accepted the service invoking request and executed the processing. If the service invoking interest comes from the unauthorized user, the computing resource of router has been occupied and wasted.

For the latter solution, it is a feasible way for NICS scenario. In this way, the user of invoking computing service will insert a signature in the interest packet during the initialization stage. The edge router or distributed NICS will verify the signature attached in this interest, then only the interest passing the verification can enter the network or executing the subsequent NICS processing. Current proposed schemes of signing interest mainly focus on the edge verification. Xue K et.al. proposed a scheme named SEAF[12], which check the validity of received interest at the network edge, then effectively reduce the data traffic in the network. Literature [13] gave a similar design. But facing the NICS characteristics of dynamically deployment and migration, edge router is difficult to maintain the authorization relationships between NICS and authorized users. Therefore, to implement interest signature-based access control for NICS, it is better to verify the signature in the location of deployed NICS.

Now we focus on the interest signature-based access control scheme with verification in router side, the core design is how to balance the verification delay and key storage cost. Usually, the authorized user will sign the service invoking interest using its private key, and if the requested NICS owns the public keys of all authorized users, NICS side can check the legality of received interest easily[14]. But unfortunately, attaching all authorized user's public keys to the VM or container image is not secure. Moreover, if the number of authorized users is large, the image size will become too heavy, then leading inconvenient deployment and migration.

Another conventional way of checking the authorized users for a specific NICS is to execute the authentication task by source server[15]. When a NICS receives the service invoking interest, it directly forwards the interest to corresponding source server. According to the returned authentication result, the NICS determines to accept the request or deny it. This way avoids the information leakage risk of authorized users and bulky image size of VM or container, but extra transmission delay caused by source server authentication will degrade the service invoking quality for authorized user.

To implement distributed authentication with low cost for NICS, Certificateless Signature is a feasible way[16], which is an improvement design of Identity Based Signature (IBS). In the certificateless signature scheme, the user's public key is bound with its ID, so the NICS side just uses the user ID to check the authenticity of requester and without stores the authorized user's public key again. But traditional certificateless signature scheme cannot fine-grained identify the subscription relationship between requester and its requested computing service, and how to dynamically control the access permission under distributed scenario is still a challenge.

Aiming at these problems, we propose a novel certificateless signature-based access control scheme (CS-ACS for short) for NICS in this paper. In CS-ACS, the source server generates a special partial public-private key pairs to the authorized user according to three parameters: the user ID, subscribed computing service name, and service expiration time. Combining with the self-generated partial public-private key pair, the authorized user gets the entire secret keys. To access the subscribed service, the authorized user signs the interest of invoking target service using its private key, while inserting its public key and expiration time into the interest packet. Based on our special signature design, for NICS side, it only needs to host the system master public key to check whether the requester is an authorized user for requested service or not. Moreover, due to that the service expiration time in our design is unforgeability, the NICS can check out the invalid request once the subscribed service expires. Simulation results show that, using our scheme, NICS can independently identify the authorized users for a specific service with excellent secret key storage cost and reasonable authentication delay.

The remainder of the paper is organized as follows. The network model and detailed designs are presented in Section 2. Section 3 evaluates our scheme through simulations and gives the security analyses. In Section 4, we summarize this paper.

II. The Design of CS-ACS

A. Network Model

The network we designed in this paper is shown in Figure.3, which consists of three entities, including source server, NDN Router and user.

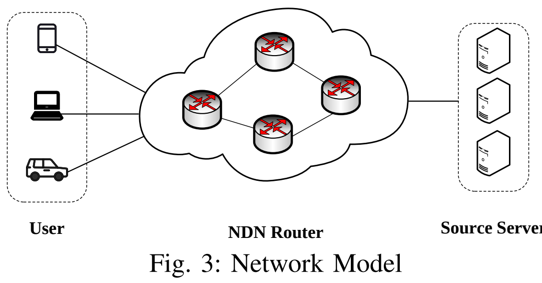

Fig. 3: Network Model

Source Server. As the service provider, source server runs specific named computing services, while encapsulating these computing services into containers and releasing the container images to the network. Furthermore, for a specific computing service, the source server side manages the subscription information of each authorized user, assigns and withdraws the secret key to/from each authorized user.

NDN Router. NDN router provides named in-network computing services through loading the container images using its free resources. To forbid the unauthorized user to invoke the computing service, the NDN router need to check the subscription relationship between the received interest packet and its requesting service.

User. Under CS-ACS, the user (terminal device) should first subscribe the target computing service by registering to source server. After obtaining legal identity, it can send interest packet to invoke named computing service from nearby NDN router or source server.

B. Detailed Description

The CS-ACS is designed based on elliptic curve cryptography, the fundamental parameter settings and notations used in CS-ACS are defined as follows: (1) We set p as a large prime; (2) G is an additive group on an elliptic curve, P is its generating element and q is its order; (3)$H : \{0,1\}^* \rightarrow Z_q^*$ is one-way hash function based on SHA-1 algorithm, where Z_q^* represents the remaining part of the integer Z_q after the 0 is removed.

The running process of CS-ACS includes three stages as shown in Figure.4.

Stage 1: Key generation and assignment

(1) System master public-private key pairs generation

In initialization stage, source server randomly selects the system master private key $k_{ms} \in Z_q^*$,which is used to calculate the system master public key through formula $P_{pub} = k_{ms} * P$. Next, source server keeps k_{ms} privately and exposes the system security parameters P_{params} to all users and NICS, here $P_{params} = \{p, q, P, P_{pub}, H\}$.

(2) User public-private key pairs generation

According to the mechanism of certificateless signature, the entire user public-private key pairs are composed of two parts, server side and user side.

When user i begins to subscribe a specific service (e.g. *Service_A*), it first randomly selects a secret value $x_i \in Z_q^*$ as its private key (user side), then uses P_{params} to calculate its public key (user side)$X_i = x_i * P$.

Next, user i sends an interest packet with X_i and its identity ID_i to source server to subscribe the target computing service.

After receiving this interest, the source server extracts the subscribed service name, user identity and user public key (user side) from it, assigns an expiration time t_i and a random number $r_i \in Z_q^*$ for this subscription,then generates the user public-private key pairs (server side) (R_i, d_i)according to the following calculation relationships.

$$\begin{cases} R_i = r_i * P \\ h_1 = H\left(ID_i \left\|R_i\right\| X_i \| \mathit{Service\_A} \| t_i\right) \\ d_i = r_i + k_{ms} * h_1 \end{cases} \tag{1}$$

Where k_{ms} is the system master private key, R_i is the user public key (server side) and d_i is the user private key (server side). In formula (1), we can find that R_i and d_i are determined by ID_i, *Service_A*, and t_i.This special design not only binds the authorized user's identity to the subscribed

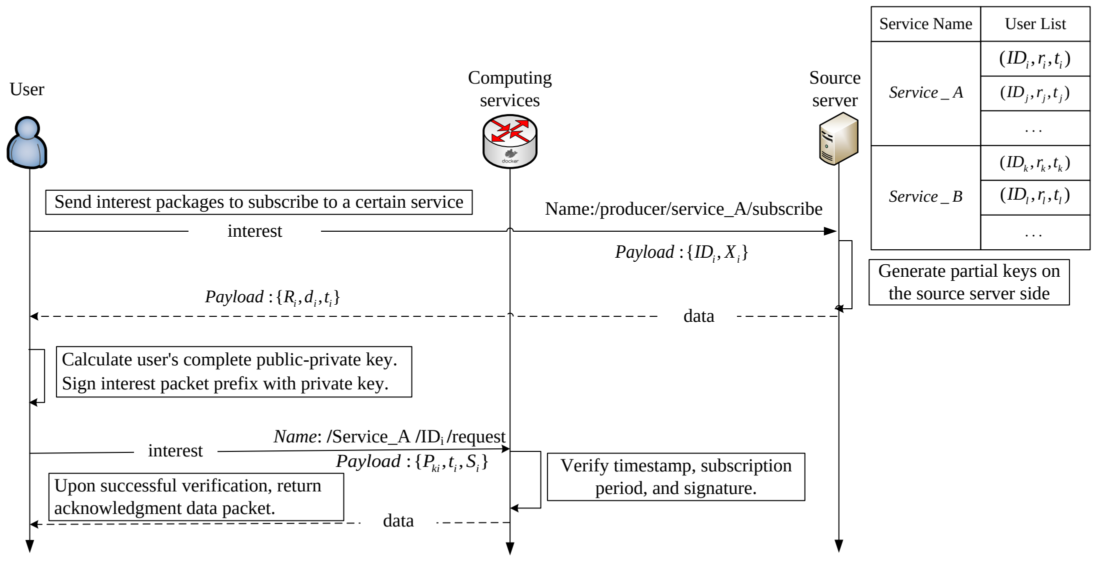
User
Computing services
Source server
Service Name
User List
Service_A
(ID_i, r_i, t_i)
(ID_j, r_j, t_j)
...
Service_B
(ID_k, r_k, t_k)
(ID_l, r_l, t_l)
...
Send interest packages to subscribe to a certain service
interest
Name:/producer/service_A/subscribe
$Payload: \{ID_i, X_i\}$
Generate partial keys on the source server side
$Payload: \{R_i, d_i, t_i\}$
data
Calculate user's complete public-private key. Sign interest packet prefix with private key.
interest
Name: /Service_A /ID_i /request
$Payload: \{P_{ki}, t_i, S_i\}$
Verify timestamp, subscription period, and signature.
Upon successful verification, return acknowledgment data packet.
data

Fig. 4: The running process of CS-ACS

service, but also associates the validity of user key with the expiration time. Next, the source server returns a data packet with R_i, d_i and t_i to user *i*, which is encrypted with X_i.

Once user *i* receives this data packet, it firstly judges the legitimacy of R_i, d_i and t_i by the equation $d_i * P = R_i + P_{pub} * H\left(ID_i \left\| R_i \right\| X_i \| \mathit{Service\_A} \| t_i\right)$. If the equation holds, user *i* further combines the received (R_i, d_i) with its local (X_i, x_i) to obtain the entire public-private key pairs. Finally, its private key is $S_{ki} = (x_{i,}, d_i)$ and its public key is $P_{ki} = (X_i, R_i)$.

Stage 2: Signing the interest packet of invoking NICS

If user *i* want to invoke *Service_A*, it generates an interest packet with name prefix "/*Service_A*/ID_i/request". Within this interest packet, user public key $P_{ki} = (X_i, R_i)$, service expiration time t_i and a signature S_i are attached. The signature rule of S_i is shown in formula (2).

$$\begin{cases} h_2 = H\left(ID_i \left\| R_i \right\| \mathit{Service\_A} \| T\right) \\ S_i = d_i + x_i * h_2 \end{cases} \tag{2}$$

Where, *T* is a timestamp which is used to prevent the replay attack. Obviously, S_i is signed by the private key of user *i*, which is used to prove user *i* owns valid subscription relationship.

Stage 3: Signature verification in NICS side

For any deployed NICS of *Service_A*, it only holds the system master public key P_{pub}. When it receives the invoking interest packet, it extracts the information of (X_i, R_i), ID_i, T and t_i from it, then verifies the signature using the following steps.

(1) Checking the timestamp *T*. If *T* is not equal to current time, the interest packet is discarded.

(2) Checking the expiration time t_i. If t_i is earlier than current time, the interest packet is discarded.

(3) Checking the signature S_i using formula (3). If $S_i * P = R_i + P_{pub} * h_1 + X_i * h_2$ is held, it is proved that this request for *Service_A* is valid, current NICS of *Service_A* will further process this invoking request; otherwise, the interest packet is discarded.

$$\begin{cases} h_1 = H\left(ID_i \left\| R_i \right\| X_i \| \mathit{Service\_A} \| t_i\right) \\ h_2 = H\left(ID_i \left\| R_i \right\| \mathit{Service\_A} \| T\right) \\ S_i * P = R_i + P_{pub} * h_1 + X_i * h_2 \end{cases} \tag{3}$$

The aforementioned mechanism of CS-ACS provides an effective way for NICS to check the subscription relationship of requester and requested service. To achieve the distributed independent verification ability, the NICS only needs to hold the system master public key P_{pub}, so CS-ACS is very friendly for NICS side.

From the security aspect, CS-ACS is secure.

(1) The user public-private key pairs aren't tampered. Due to that the system master private key k_{ms} is hidden to the user, user cannot tamper its public-private key pairs (server side) (R_i,d_i) .

(2) The sensitive information associated with user identification and subscription service aren't tampered. Due to that the user private key (server side) is determined by user identity ID_i , subscribed service name *Service_A* and assigned expiration time t_i, if the user viciously changes any above sensitive information, the signature signed by its unforgeable private key cannot pass the NICS side verification. For example, once t_i is expired, the authorized user must re-subscribe the computing service and obtain a new user public-private key pairs (server side) from source server. Otherwise, if the user modifies the expiration time t_i locally, the NICS side will use the received fake expired time to calculate h_1 of formula (3), then the signature verification will fail. This design is benefitted to NICS to control the user permission.

(3) The signature can defend the replay attack. Due to the signature S_i is generated from the concatenated string h_2 which consists of timestamp *T*, the NICS side will use current time to check the received signature according to formula (3). So, even if a valid signature is hijacked by the unauthorized user to invoke one computing service, it cannot pass the verification of NICS side.

III. Performance Evaluation

In this section, we use ndnSIM (version 2.3)[17] to evaluate the performance of CS-ACS from four aspects, average hit probability of NICS, average NICS invocation delay, the storage cost on NICS side, and the average authentication delay. The comparison schemes selected in our simulations are

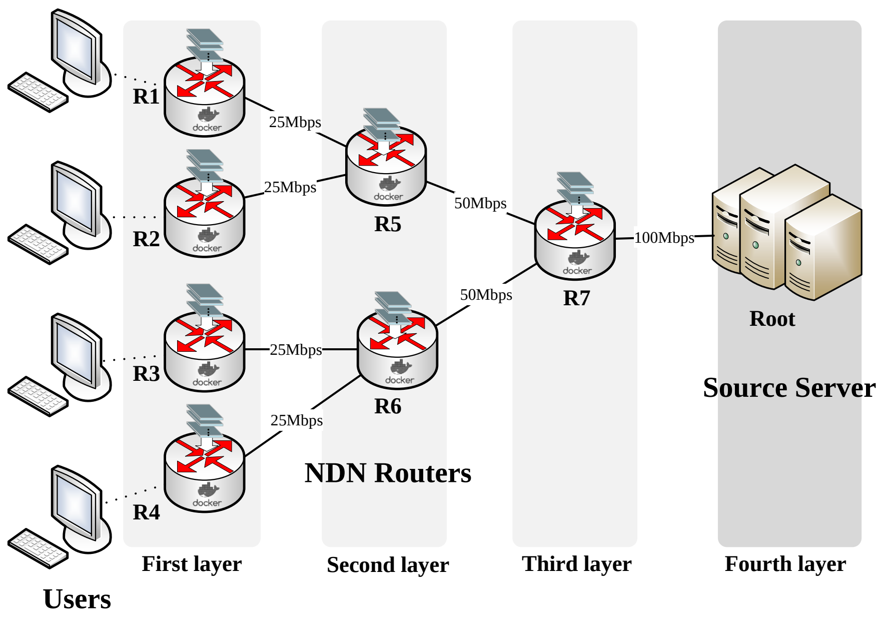

Fig. 5: Simulation Topology

two fundamental solutions mentioned in section 1, we name them as ECDSA_Docker and ECDSA_Server respectively. Both two comparison schemes adopt elliptic curve digital signature algorithm (ECDSA) as their signature algorithm, but the former verifies the requester identity on NICS side (NICS container stores the public keys of all authorized users), the latter doesn't execute the authentication work on NICS side and just forwards the interest to source server.

The simulation settings are as follows.

(1) Network topology is shown in Figure.5, which consists of 4 users, 7 NDN routers and 1 source server.

(2) The source server provides 10 classes computing services, the popularity of computing services obeying Zipf distribution where the service class 1 owns highest popularity. In simulation, we adopt three Zipf distribution parameters ($\alpha = 0.8$, 1, 1.2) to disclose the affecting from the service popularity.

(3) For each NDN router, its computation resource is limited. We assume that one router only loads maximum 3 or 7 computing service container simultaneously (the container of a specific service can be repeatedly deployed).

(4) The delay of each hop is set as 10 ms. For achieving the fair evaluation, in simulations, we set the public key size is 160 bits in three schemes.

From the network performance with NICS deployment, Figure 6-Figure 8 show the NICS hit probability and average invocation delay in the network. In the simulation experiments of Figure 6 and Figure 7, we set the NICS invoking rate as 100 times/second; in the simulation experiments of Figure 8, we change the NICS invoking rate from 8000 times/second to 12000 times/second.

Obviously, with the increasing of deployed NICS number, the hit probabilities of all classes are improved, especially for the NICS of high popularity, as shown in Figure 6. On the other hand, for the routers located at the second layer and the third layer, the NICS of high popularity running on them achieve low hit probability than that of low popularity. This phenomenon is totally different from the routers of first layer. The reason is that when the NICS invoking rate is low, the major NICS invoking requests will be satisfied in the first layer and only the rest unhit requests will be forwarded to the second and the third layers. Figure 7 show the average hit probability of four layers, the simulation result also disclose the effect of the popularity, deployed NICS number and the router location.

Figure 8 gives the NICS invocation delay with the increasing invoking rate. As shown in Figure 8, due to that the each NICS task will consume a specific time to execute, the processing capacity of each router is limited, it is different from static content hit event. So, when the invoking rate from users become high, more invoking requests will be forwarded to next hop to execute, then leading to the rapid increased invoking time.

The results of Figure 6-Figure 8 show that more deployed NICSs will promote the network performance, high popularity NICS will achieve more hit probability in the network. But the network processing capacity exists the upper limitation, if the invoking rate become very high, most requests will be forwarded to the source server to execute.

From the security aspect, Table 1 gives the storage cost on NICS side under three schemes. For ECDSA_Docker, it needs to store all public keys of authorized users. If we assume n is the authorized user number, the storage cost of ECDSA_Docker shows linear relationship with n. For ECDSA_Server, due to that it doesn't execute local verification, its storage cost is zero. For CS-ACS, it only needs to host the system master public key, so its storage cost just equals to the size of elliptic curve public key and has negligible influence on the container running NICS.

TABLE I: Storage Cost on NICS Side.

Scheme	ECDSA_Server	ECDSA_Docker	CS-ACS
Storage Cost(bit)	None	$160n$	160

Figure.9-Figure.13 show the simulation results of average invocation delay under three schemes, where the invocation delay is defined as the total time from generating the interest signature to receiving the NICS acknowledgement

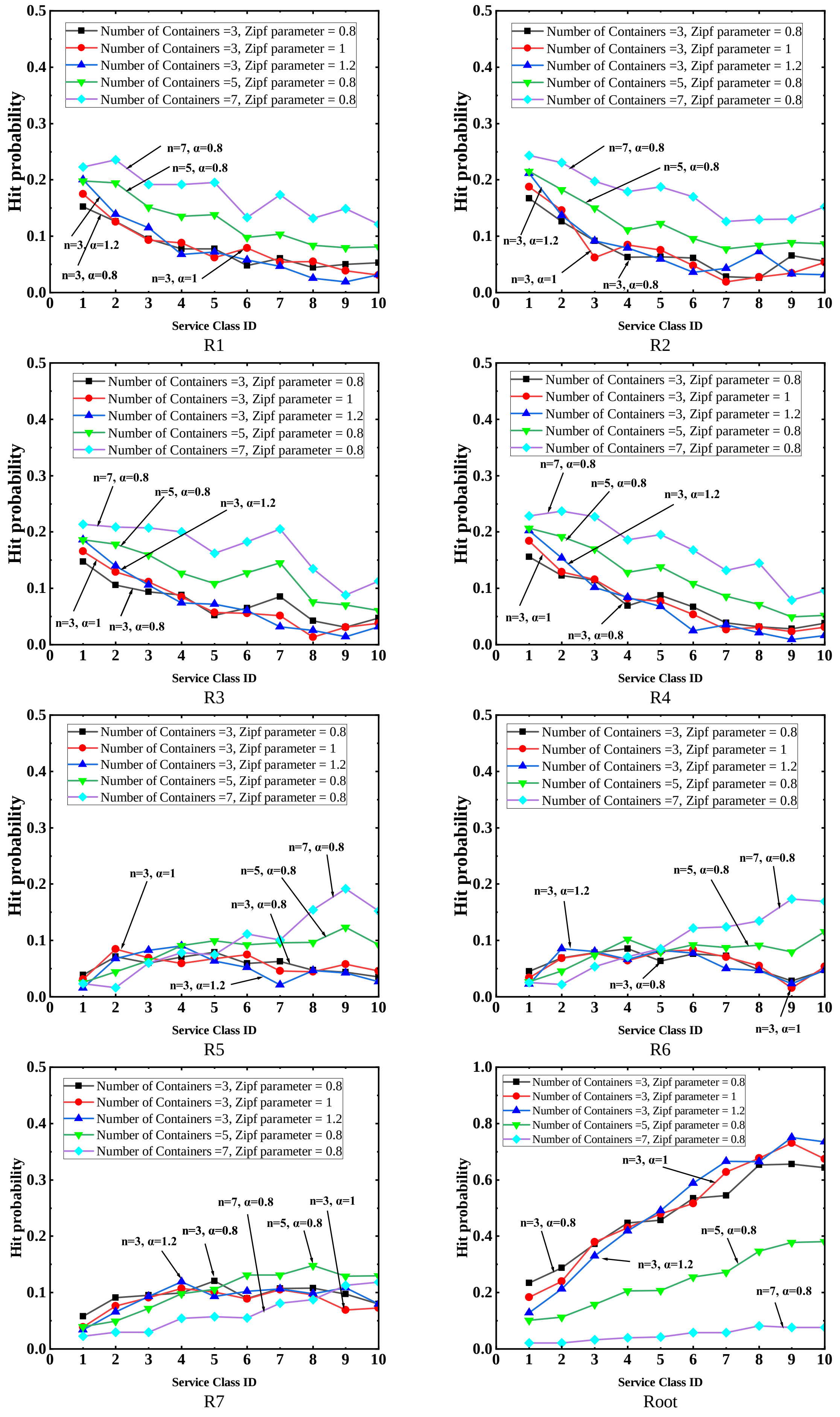

Fig. 6: Computing service hit probability on each router

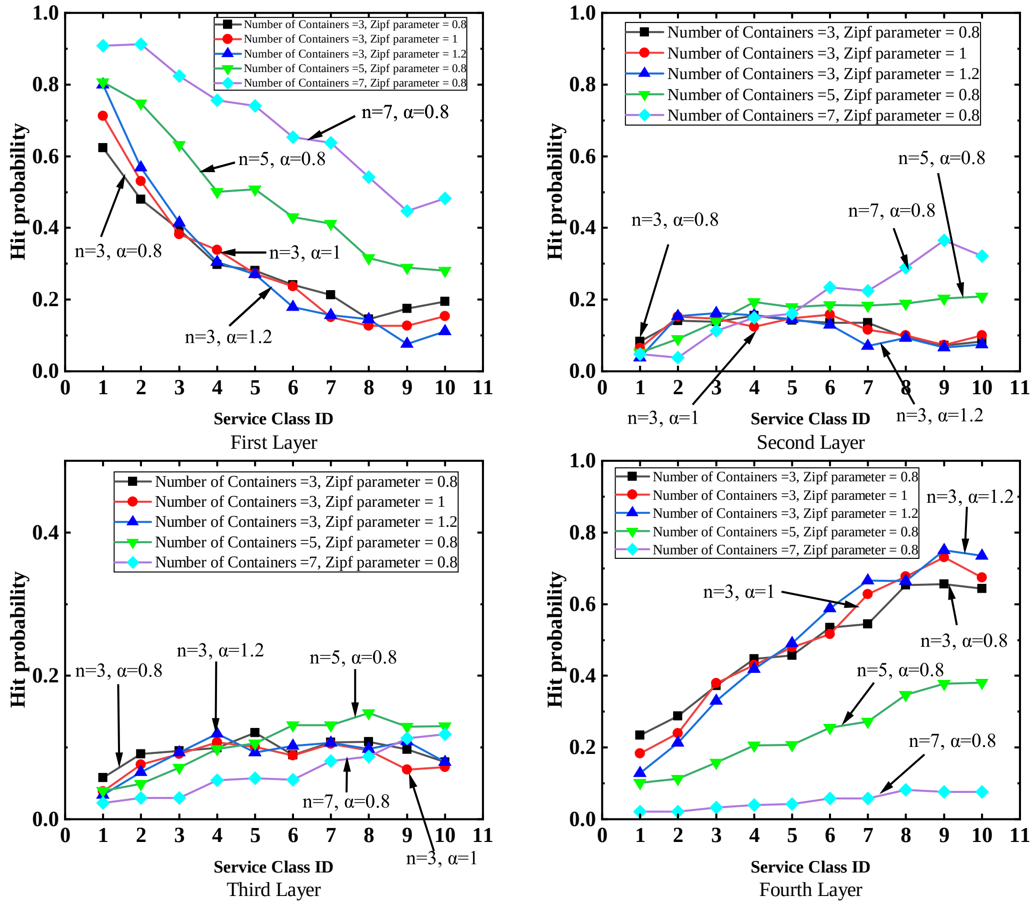

Fig. 7: Computing service hit probability in different layers

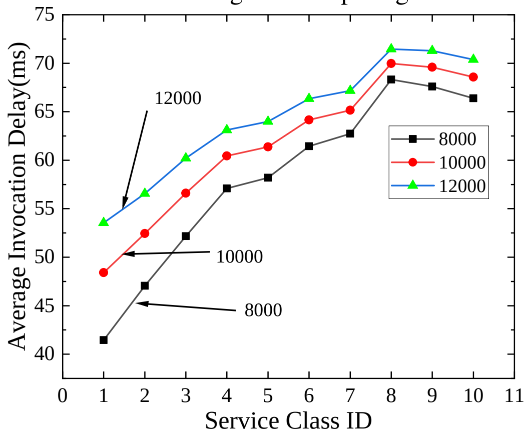

Fig. 8: The service invocation delay at different request rates

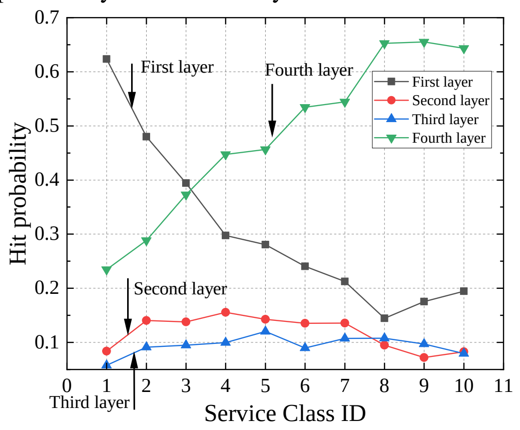

Fig. 9: Computing service hit probability in different layers

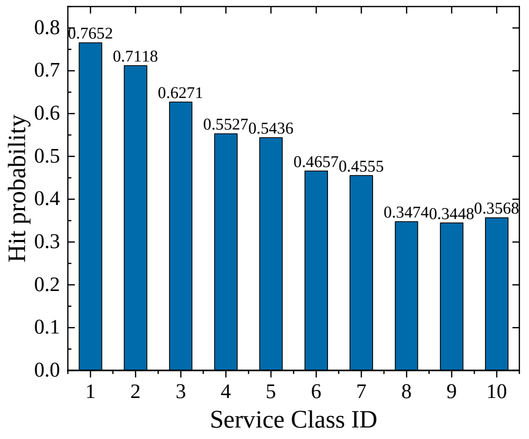

Fig. 10: Total in-network hit probability of top 10 classes

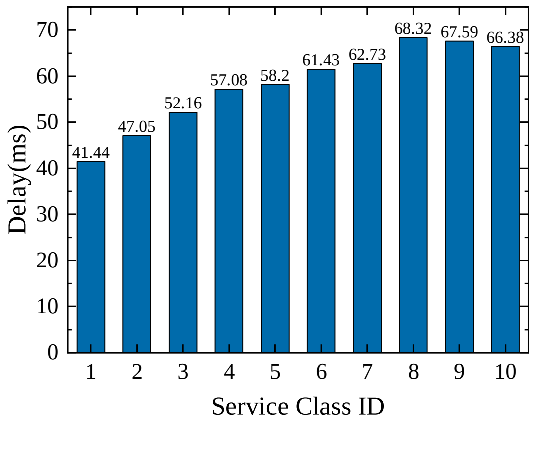

Fig. 11: Average RTT of top 10 classes

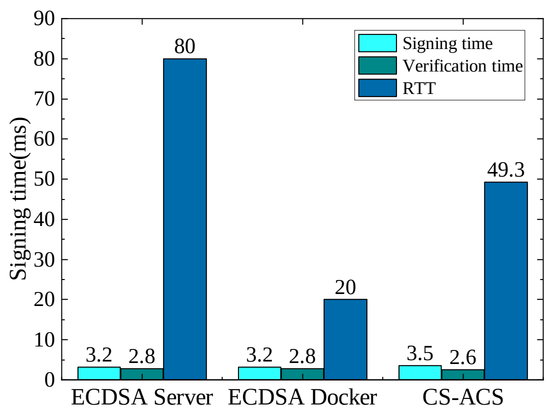

Fig. 12: Average signing, verification time and RTT.

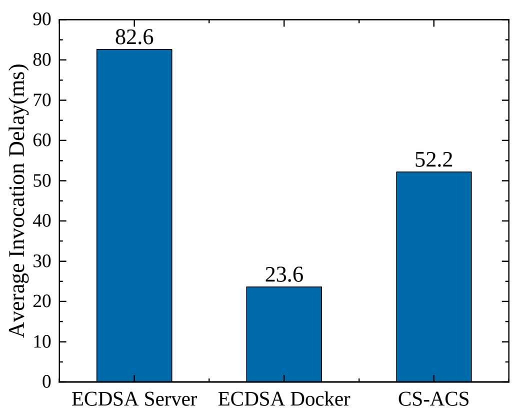

Fig. 13: Total authentication delay.

data. It consists of three parts, average signing time, average verification time, round trip time (RTT) between user and hit NICS location. The results are counted by 3000 random simulations.

Figure.9 gives the computing service hit probability in different layers. Similar to the cached content hit event, the computing service with high popularity is also easy to hit in the routers close to user. As shown in Figure 9, for the computing service of class 1, around 65% invoking requests are satisfied in the first layer routers, the rest requests are scatteringly satisfied in the second to fourth layers. Figure 10 gives the total in-network hit probability of top 10 classes, obviously higher in-network hit probability is, lower RTT between user and hit NICS location is. Figure 11 clearly shows the average RTT of top 10 classes, for the class 1 and class 10, there is around 25ms RTT difference. Figure.12 and Figure.13 further show the comparison of average invocation delay.

From the results, we can see the time costs of CS-ACS during signing and verification stage are both slightly higher than other two schemes. The reason is that our design increases the complexity of signing and verification algorithm. But from the aspect of total authentication delay, CS-ACS and ECDSA_Docker execute local verification, avoid the forwarding round trip time between source server and itself, so they have distinct advantage over ECDSA_Server. From the aspect of storage cost on NICS side, the advantage of CS-ACS and ECDSA_Server are clear than ECDSA_Docker. So, CS-ACS owns the best overall performance.

IV. Conclusion

Facing the requirement of distributed authentication under NICS scenario, we propose a novel certificateless signature-based access control scheme in this paper. Based on a special design of user public-private key pairs (server side), we realize three features, including (1) NICS can verify the signature embedded in the interest packet just using the system master public key; (2) NICS can check out the subscription relationship between requester and requested computing service; (3) NICS can check out the subscription relationship is expired or not. Simulation results show that, comparing with two fundamental solutions, our scheme can achieve excellent performance in both aspects of storage cost and authentication delay.

References

[1] Zhang L, Afanasyev A, Burke J, et al. Named Data Networking. ACM SIGCOMM Computer Communication Review. 2014; 44(3): 66–73.
[2] Huang S, Chen R, Li Y, et al. Intelligent Eco Networking (IEN) III: A Shared In-network Computing Infrastructure towards Future Internet. In: 2020 3rd International Conference on Hot Information-Centric Networking (HotICN); 2020: 47-52.
[3] Tong Sun, Chao Ma, Zechun Li, and Kun Yang. Cloud Computing-based Parallel Deep Reinforcement Learning Energy Management Strategy for Connected PHEVs. Engineering Letters. 2024; 32(6):1210-1220.
[4] Krol M, Psaras I. NFaaS: Named Function as a Service. In Proceedings of the 4th ACM Conference on Information-Centric Networking (ICN 2017); 2017: 134-144.
[5] Saurabh Singhal, and Ashish Sharma. Mutative ACO based Load Balancing in Cloud Computing. Engineering Letters. 2021; 29(4): 1297-1302.
[6] Fang P, Wolf T. Enabling Virtual Network Functions in Named Data Networking. In: IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS); 2021: 1-6.
[7] Kondo D, Ansquer T, Tanigawa Y, Tode H. Resource Discovery for Edge Computing over Named Data Networking. In: Proceedings International Computer Software and Applications Conference. 2021: 552-559.
[8] Odun-Ayo I, Omoregbe N, Udemezue B. Cloud and mobile computing - Issues and developments. Lecture Notes in Engineering and Computer Science. 2018: 363-368.
[9] Isaac Odun-Ayo, Olasupo Ajayi, and Sanjay Misra. Cloud Computing Security: Issues and Developments. Lecture Notes in Engineering and Computer Science. 2018: 175-181.
[10] Krol M, Marxer C, Grewe D, Psaras I, Tschudin C. Open Security Issues for Edge Named Function Environments. IEEE Communications Magazine. 2018; 56(11): 69-75.
[11] Meirovitch D, Zhang L. NSC–Named Service Calls, or a Remote Procedure Call for NDN. tech. rep., NDN Project, Technical Report NDN-0074; UCLA Computer Science Department: 2021.
[12] Xue K, He P, Zhang X, et al. A Secure, Efficient, and Accountable Edge-Based Access Control Framework for Information Centric Networks. In IEEE/ACM Transactions on Networking. 2019; 27(3): 1220-1233.
[13] Tourani R, Stubbs R, Misra S. TACTIC: Tag-Based Access ConTrol Framework for the Information-Centric Wireless Edge Networks. 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS). 2018: 456-466.
[14] Tao Y, Zhu Y. An interest-based access control scheme via edge verification in Named Data Networking. International Journal of Communication Systems 2022; 35(10).
[15] Zhang Z, Yu Y, Afanasyev A, Burke J, Zhang L. NAC: Name-Based Access Control in Named Data Networking. In: Proceedings of the 4th ACM Conference on Information-Centric Networking (ICN 2017); 2017: 186-187.
[16] Barbosa M, Farshim P. Certificateless Signcryption. In: Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security; 2008: 369–372.
[17] Mastorakis S, Afanasyev A, Zhang L. On the Evolution of ndnSIM: an Open-Source Simulator for NDN Experimentation. ACM SIGCOMM Computer Communication Review; 2017; 47(3): 19-33.