scientific reports

Check for updates

OPEN

Security verification framework for NDN access control

Yuan Fei[1,2,4], Jiaqi Yin[3,4] & Lijun Yan[1]✉

Named Data Networking (NDN) presents a promising alternative to TCP/IP, but its access control design poses challenges for cybersecurity. Addressing this, the paper introduces the Security Verification Framework for NDN Access Control (SVF-NDN). This framework employs formal analysis to assess access control schemes, evaluating their resilience against cyberattacks. SVF-NDN verifies five crucial security properties-deadlock freedom, data availability, key authentication, data leakage protection, and data access protection. Implemented using the PAT model checking tool, the framework focuses on a data encryption-based NDN access control. Uncovering vulnerabilities such as node key pair faking and data leakage, two enhancement methods are proposed and evaluated. Recognizing the potential compromise of Access Control Manager (ACM), an innovative solution is presented. Additionally, four algorithms streamline the automatic updating of formal models. Results indicate SVF-NDN's efficacy in fortifying access control against cyber threats, offering valuable insights for bolstering NDN security.

Named Data Networking (NDN)[1]is one of the leading architectures in Information-Centric Networking (ICN) that aims to resolve the existing problems in TCP/IP Internet[2,3]. Although TCP/IP-based network has shown great resilience over the years, it cannot support the newly evolving content distribution model successfully, as users gradually pay more attention to named content rather than its location. NDN emerges as one of the promising architectures in ICN, where each packet does not carry an IP address but a data name. When a data consumer needs data, it sends out an *Interest* packet with the required name of the data. According to the name, routers forward the packets over the network, and a *Data* packet with a matching name will be returned to the consumer when it is produced by some data producer.

Although the architecture of NDN is different from traditional networks, it also faces cyberattacks. Access control is an important measure to protect network security. As a fundamental aspect of network security, it is strongly correlated with other security services such as authenticity, auditing, and authorizations[4]. Generally, the main purpose of access control is to regulate who can view or use resources in a computing environment. Traditional mechanisms of access control focus on the IP addresses of end hosts. Such host-centric access control models cannot be easily adapted into NDN. As *Data* packets are cached in the content store at NDN routers for effective data delivery, they may be obtained by the consumers without access rights.

Some solutions have been proposed for ICN architectures including NDN with several limits. Data encryption is a natural and intuitive approach to build access control. Chen et al.[5] proposed an encryption and probability-based access model for NDN. The bloom-filter data structure applied in this model is suitable for video streaming services, but may reduce the efficiency in other scenarios. In Misra et al.[6], contents are encrypted by a symmetric data key whose dissemination is supported by Broadcast Encryption (BE). However, BE is only suitable for the context where the number of users is limited. As the prototype of NDN, CCNx introduced a simple access control solution[7], which allows the control of the rights of reading, writing, and management. Unfortunately, its lazy revocation produces a possible situation where a revoked entity reads protected content. To address these limits, Hamdane et al.[8] introduced a new encryption-based NDN access control, which is also the research object of this paper.

At present, NDN access control design is still in the starting stage, and it faces a lack of systematic guidance and the ability to adjust for cyberattacks. Most of them lack the verification of the security properties related to attacks and need more flexible measures for different cyberattacks. As a method that can describe the system and its requirements formally, formal verification (e.g., model checking) is helpful to support verification of the security properties related to cyberattacks. Model checking helps to avoid manual testing and save time. At the same time, it supports the feature of giving counter-examples, which can effectively provide clues for the

[1]College of Information, Mechanical and Electrical Engineering, Shanghai Normal University, Shanghai 201418, China. [2]State Key Laboratory of Systems Medicine for Cancer, Center for Single-Cell Omics, School of Public Health, Shanghai Jiao Tong University School of Medicine, Shanghai 200025, China. [3]School of Software, Northwestern Polytechnical University, Xi'an 710129, China. [4]Yuan Fei and Jiaqi Yin have equally contributed to this work. ✉email: flying@shnu.edu.cn

improvement of access controls to enhance their robustness to cyberattacks. Therefore, formal verification is very recommended for protecting NDN from cyberattacks.

As one of the most popular formal verification techniques, model checking is suitable for modeling access control schemes[9]. However, model-checking-based methods are still not widely used in this field. One reason is that it is very difficult for users to build a formal model directly from the access control. Although some researchers have conducted formal verification of their access control schemes[10,11], there is a lack of methods to support NDN access control. To solve this problem, it is demanded that the formal verification method should generate formal models from the NDN access control as automatically as possible.

Contributions

As discussed above, the motivation is to provide a security verification framework for NDN access control (SVF-NDN). Fig. 1shows the overall framework of SVF-NDN. It supports the CSP (Communicating Sequential Processes) modelling from the NDN access control[8] and formalizing security properties into linear temporal logic (LTL) formulas, which are verified by model checker PAT (Process Analysis Toolkit). The properties include deadlock freedom, data availability, key authentication, data leakage prevention, and data access protection. SVF-NDN supports updating NDN access control based on verification results.

This paper extends the previous work published at ICFEM 2018[12]. In[12], the NDN access control was modeled, and four properties (deadlock freedom, data availability, key authentication, and data leakage protection) were verified. Additionally, two model improvements were presented. Now, to provide a more comprehensive description of various scenarios, additional types of messages are introduced, and a new property, namely data access protection, is verified. In the first model improvement, compared to the approach presented in[12], additional modeling components for different entities and intruders have been incorporated. Furthermore, three new algorithms (Algorithm 1, Algorithm 2, and Algorithm 3) have been introduced to facilitate the updating of CSP models. In the second model improvement, compared to the one presented in[12], additional modeling components for different entities and intruders have been incorporated. Furthermore, a new algorithm (Algorithm 4) has been introduced. Algorithm 4 is introduced to support updating CSP overall models. Considering the situation when ACM is invaded, another new solution (an innovative method) is introduced, which is also modelled and verified.

Results

NDN access control

Hamdane et al.[8] proposed an access control solution for NDN based on data encryption. Symmetric data key *DK* is used to encrypt the data produced by a writer. A pair of keys ($NK_{encryption}$,$NK_{decryption}$) is specially used to encrypt *DK*. Similar to public and private keys, $NK_{encryption}$ is used for encryption and $NK_{decryption}$ does the decryption job. However, both of them are secret. There are three entities in this scheme. The establishment of this access control solution is mainly based on the entities as below:

- **Readers and writers:** They correspond to users with the read and write rights respectively. Readers want to read encrypted data. Writers are responsible for producing encrypted data.
- **Access Control Manager (ACM):** It is introduced to control the management of the access control policy.

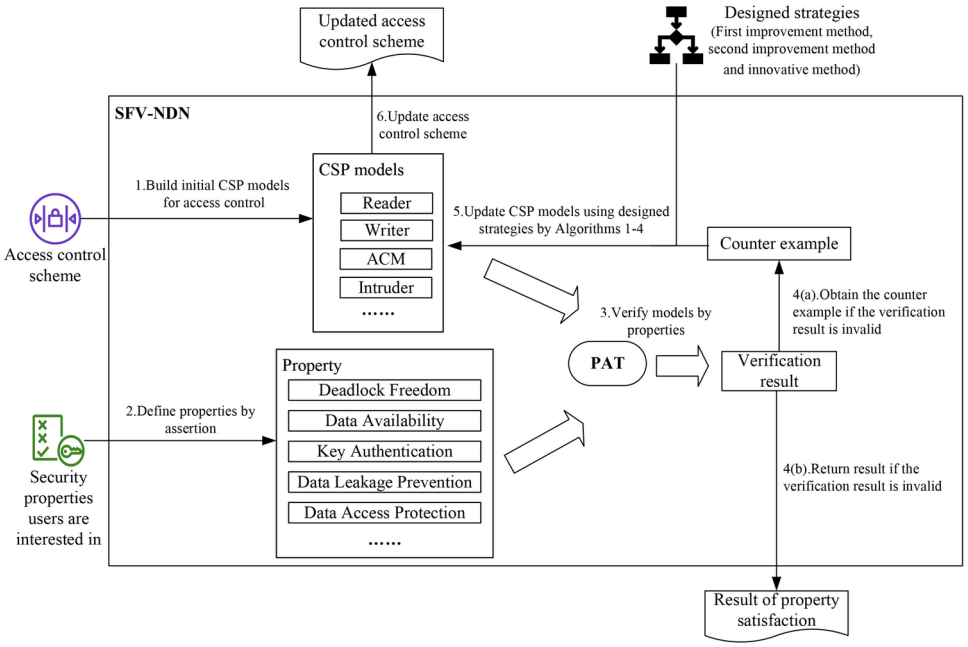

Fig. 1. Overall framework of SFV-NDN.

- **Network Nodes (NN):** They guarantee content delivery to transit messages between entities and ACM.Since read and write operations in closed environments are the most complex ones in[8], they are chosen to be studied in this paper. The focus of this research is solely on the transmission of key information, including keys and data. Therefore, several operational steps, such as checking the naming convention and the writing privilege of the writer, among others, have been omitted. Fig. 3 illustrates simplified NDN access control with read and write operations. As *NN* is only responsible to transit message to *ACM*, Reader_i and Writer_i are communicating with *ACM* essentially.
-
- **Read Operations**: To reduce the size of the CSP models in SVF-NDN, commands and the information that the reader can get from the commands in original access control are abstracted into an *Interest* packet and a *Data* packet respectively. The *Interest* packet confirms whether the reader has the privilege to read and whether the namespace where the data name belongs owns the privilege to read (step a.0). The *Data* packet is used to inform the reader about the name of the node key and the data key (step a.0'). As a reader requests the desired content (step a.1), he gets the encrypted data (step a.2). He uses special commands to get the naming conventions, transmits a request for the data key DK (step a.3), and receives it encrypted by $\mathrm{NK}_{encryption}$ (step a.4). He applies special commands to request the naming convention, and also retrieves the set of hash values of the public keys to confirm that he owns the read privilege. The reader sends the name of the node keys and the hash value of his public key to get these keys (step a.5). On receiving the response (step a.6), he can apply his public key to decrypt the received key(s). He uses $\mathrm{NK}_{decryption}$ to decrypt the data key DK and decrypt the content with this key.
- **Write Operations**: Step b.0 and step b.0' are presented similarly to step a.0 and step a.0' for reducing the CSP models. If the writer has the privilege of writing, he can continue to send the request, otherwise, it will be rejected by ACM. To request the node keys, he sends the name of the node keys along with the hash value of his public key (step b.1). Upon receiving the response (step b.2), he decrypts the pair of keys. The writer then creates two packages. The first one contains the content encrypted by data key DK, which is generated by the writer randomly. The second one includes the decryption of this DK which is done by $\mathrm{NK}_{encryption}$. After getting the permit for writing the namespace from ACM, he also receives *Interest* packets for the corresponding two packets (steps b.3 and b.5). The writer feeds back with the two packets. After receiving the messages (steps b.4 and b.6), the ACM saves them in the repository permanently.

Communicating sequential processes

As one of the most mature formal methods, Communicating Sequential Processes (CSP)[13,14]is tailored for describing the interaction between concurrency systems by mathematical theories. Because of its well-known expressive ability, CSP has been widely used in many fields[15–18].

CSP processes are constituted by primitive processes and actions. The following syntax is used to define the processes in this paper, whereby P and Q represent processes, the alphabets $\alpha(\mathrm{P})$ and $\alpha(\mathrm{Q})$ mean the set of actions that the processes P and Q can take respectively. Hence, a and b denote the atomic actions, and c stands for the name of a channel.

$$\mathrm{P,Q}:: = \ldots \mid \mathrm{SKIP} \mid \mathrm{a} \rightarrow \mathrm{P} \mid \mathrm{c?x} \rightarrow \mathrm{P} \mid \mathrm{c!e} \rightarrow \mathrm{P} \mid \mathrm{P;Q} \mid$$
$$\mathrm{P} \lhd \mathrm{B} \rhd \mathrm{Q} \mid \mathrm{P} \Box \mathrm{Q} \mid \mathrm{P}[[\mathrm{a} \leftarrow \mathrm{b}]] \mid \mathrm{P}[|\mathrm{c}|]\mathrm{Q}$$

Here, SKIP stands for a process that only terminates successfully. $\mathrm{a} \rightarrow \mathrm{P}$ first performs action a, then behaves like P. $\mathrm{c?x} \rightarrow \mathrm{P}$ receives a message by channel c and assigns it to variable x, then does the subsequent behavior like P. $\mathrm{c!e} \rightarrow \mathrm{P}$ sends a message e through channel c, then performs P. P;Q executes P and Q sequentially. $\mathrm{P} \lhd \mathrm{B} \rhd \mathrm{Q}$ denotes if the condition B is true, the process behaves like P, otherwise, like Q. $\mathrm{P} \Box \mathrm{Q}$ acts like either P or Q and the environment decides the selection. $\mathrm{P}[[\mathrm{a} \leftarrow \mathrm{b}]]$ changes event a to event b. $\mathrm{P}[|\mathrm{c}|]\mathrm{Q}$ indicates that P and Q execute the concurrent events on the set c of channels.

Verification for properties of initial model for NDN access control

As a model checking tool, Process Analysis Toolkit (PAT)[19]is designed as an extensible and modularized framework based on CSP. Different model checking techniques are implemented in PAT, supporting many assertions, such as deadlock freeness and reachability[20]. PAT has been applied in various places[20,21]. With advanced optimization techniques implemented in PAT, it can achieve good performance. In this section, the five properties (deadlock freedom, data availability, key authentication, data leakage protection, and data access protection) are verified with the help of the model checker PAT.

To perform the verification, the formal models have been implemented in PAT. According to the verification results, the models is also improved twice for better safety performance. The datasets utilized in the verification process essentially originate from multiple CSP models constructed from the access control scheme. In this case study, the access control scheme being considered is NDN access control. Hence, the datasets correspond to the NDN access control scheme.

Properties

Five properties of the models are verified, including deadlock freedom, data availability, key authentication, data leakage protection, and data access protection. Some of them are described in Linear Temporal Logic (LTL) formula, which is commonly used to describe linear-time properties. Because the five properties will be verified for all the models in this paper, *System()* are used to represent the models. PAT supports the LTL formula by

using the assertion #assert $P()$ |= F to check whether system $P()$ satisfies the LTL formula F. In addition, PAT provides reachability checking with the keyword "reaches".

Property 1: Deadlock Freedom

#assert System() deadlockfree;

It must be guaranteed that the models should not run into a deadlock state. PAT owns a primitive to describe this situation.

Property 2: Data Availability

#define Data_Acquisition_Success
data_acquisition_success==true;
#assert System() reaches Data_Acquisition_Success;

This assertion is a reachability property, which means the model can reach a state at which the given condition is satisfied. The situation in which the data can be transmitted to the entity requiring it is described.

Property 3: Key Authentication

#define NK_Faking_Success
nk_faking_success==true;
#assert System() |= []!NK_Faking_Success;

Once the NK key pair is faked, other security issues may appear. So this assertion is used to check whether the NK key pair can be faked successfully, using the "always" operator [] in LTL.

Property 4: Data Leakage Prevention

#define Data_Leakage_Success
data_leakage_success==true;
#assert System() |= []!Data_Leakage_Success;

The security of data should be maintained, as the leakage of data will produce a bad effect. The assertion is built to check whether the data can be obtained by intruders.

Property 5: Data Access Protection

#define Revoked_Entity_Denied_Success
((*cuEn==reEn*) &&(*is_rejected==true*))
||((*cuEn*!=*reEn*) &&(*is_rejected==false*));
#assert System() |= [] <> Revoked_Entity_Denied_Success;

The variables *reEn* and *cuEn* represent the id of revoked entity and the current entity respectively. Using the "eventually" operator <> in LTL together with the "always" operator [], whether the system can always eventually reach one of the two states or not can be checked. One is the current entity is not revoked and its request is not rejected by ACM. The other is that the current entity is a revoked one, then its request is rejected by ACM. By utilizing the proposed algorithms, it is possible to determine whether the current entity has been revoked, and subsequently ascertain if their read or write request is rejected by the ACM.

In comparison to the previous verification[12], there have been some changes in the four properties being verified. Originally, there were three properties, but now the name of "Data Security" has been revised to the more appropriate "Data Leakage Prevention", and a new property called "Data Access Protection" has been introduced. Meanwhile, only the CSP models involving Intruder modeling are kept to enhance simplicity. This means that all the models previously labeled with an "*I*" subscript in[12] have been modified to remove the subscript. For example, a model previously known as $\mathrm{SystemR}_I$ is now referred to as *SystemR*. Furthermore, the incorporation of scenarios involving entity revocation has led to the inclusion of six new models, each denoted by a subscript "*re*", such as $\mathrm{SystemR}_{\mathrm{re}}$.

The verification results in Table 3 indicate that model *SystemR*, *SystemW*, $\mathrm{SystemR}_{\mathrm{re}}$ and $\mathrm{SystemW}_{\mathrm{re}}$ satisfy the property of deadlock freedom. This means that the four models will not run into a deadlock state.

The verification results of data availability are valid for *SystemR* and *SystemW*. Thus, the models can assure that the entity owning privilege can require its desired data. The verification results of data availability for $\mathrm{SystemR}_{\mathrm{re}}$ and $\mathrm{SystemW}_{\mathrm{re}}$ are invalid. These illustrate that the entity can no longer access the data once it is revoked.

The verification results of key authentication for *SystemR* and *SystemW* are invalid, which indicates that the node key pair can be faked successfully by intruders. As $\mathrm{SystemR}_{\mathrm{re}}$ and $\mathrm{SystemW}_{\mathrm{re}}$ have no transmission of key information, the property of key authentication is satisfied for them. For the verification results of data leakage protection, *SystemR* is valid and *SystemW* is invalid, which means the intruder can get the data in the write operation. Because $\mathrm{SystemR}_{\mathrm{re}}$ and $\mathrm{SystemW}_{\mathrm{re}}$ have no transmission of a data packet, the property of data leakage prevention is valid for them. For data access protection, all of them are valid. It illustrates that the revoked entity has no privilege to access the data.

Verification results for the first improvement method of SFV-NDN

The verification results of the first model improvement are illustrated in Table 3. The six models with subscript Sig belong to the results of the first improvement. For $\mathrm{SystemR}_{\mathrm{Sig}}$ and $\mathrm{SystemW}_{\mathrm{Sig}}$, all the verification results of them are valid, which indicates the first improvement prevents the intruder to give the fake node key pair to the entity and get the data successfully. For $\mathrm{SystemR}_{\mathrm{Sig\_re}}$ and $\mathrm{SystemW}_{\mathrm{Sig\_re}}$, their verification results are all valid except for data availability. The results mean the revoked entity can no longer access the data. For key authentication, the invalid results of $\mathrm{SystemR}_{\mathrm{Sig\_C}}$ and $\mathrm{SystemW}_{\mathrm{Sig\_C}}$ indicate that the node key pair is risky of being faked when ACM's public key known by the entity is faked by intruder's public key. For data access protection, the invalid result of $\mathrm{SystemW}_{\mathrm{Sig\_C}}$ shows that data will be leaked when ACM's public key known by the writer is faked by the intruder's public key.

Verification results for the second improvement method of SFV-NDN

The verification results of the second model improvement are illustrated in Table 3. The six models subscripted by *Dig* belong to the results of the second improvement. Except for data availability of $\mathrm{SystemR}_{\mathrm{Dig\_re}}$ and $\mathrm{SystemW}_{\mathrm{Dig\_re}}$, the validation results for the six models are valid on all properties. The verification results for $\mathrm{SystemR}_{\mathrm{Dig\_re}}$ and $\mathrm{SystemW}_{\mathrm{Dig\_re}}$ means once the entity can no longer access the data if it is revoked. The verification results are all valid for $\mathrm{SystemR}_{\mathrm{Dig}}$, $\mathrm{SystemW}_{\mathrm{Dig}}$, $\mathrm{SystemR}_{\mathrm{Dig\_C}}$ and $\mathrm{SystemW}_{\mathrm{Dig\_C}}$, which indicates that the second improvement protects data from being leaked and the node key from being faked successfully. The second improvement method does ensure the security of keys and data even if the reader or writer is invaded. However, it is based on the premise that ACM is not invaded. Once ACM is invaded, data is at risk of being faked. Specifically, when ACM receives data from the writer, the data is replaced with fake data which will be sent to the reader. Thus, another approach to solving this problem is considered in the following.

Verification results for the innovative method of SFV-NDN

Table 4 presents the verification results of the innovative method. As the new access control no longer uses node key pairs, it needs to verify the deadlock freedom, data availability, data leakage protection, and data access protection. The verification results of *System_New_re* show that the new access control can prevent the revoked entity from getting the data. All valid results of *System_New* indicate that the new access control can protect the data from being leaked.

In this study, SVF-NDN framework was employed to evaluate the security properties of NDN access control, encompassing several key aspects. Deadlock Freedom was confirmed across all models, including both baseline and improved versions, ensuring the system would not halt execution. For Data Availability, baseline models such as *SystemR* and SystemW guaranteed access for entities with proper privileges, but models with revoked entities, such as $\mathrm{SystemR}_{\mathrm{re}}$, $\mathrm{SystemW}_{re}$, and *System_New_re*, failed to uphold this property, indicating successful restriction of data access for revoked entities. Regarding Key Authentication, baseline models (*SystemR* and *SystemW*) demonstrated vulnerability to intruder attacks capable of forging node key pairs, a problem resolved in enhanced models like $\mathrm{SystemR}_{\mathrm{Sig}}$ and $\mathrm{SystemW}_{\mathrm{Sig}}$. For Data Leakage Prevention, baseline models such as *SystemW* exhibited vulnerabilities during write operations, while improved models effectively prevented such leakage. *System_New* and *System_New_re* also successfully prevented data leakage, demonstrating robustness against intrusions. Lastly, all models satisfied Data Access Protection, affirming that revoked entities were denied data access as intended.

Discussion

The results of the SVF-NDN framework highlight its robustness in verifying and improving NDN access control against cyber threats, with several key observations and insights derived from the study. The first improvement method introduced digital signatures, successfully addressing key authentication and data leakage issues observed in the baseline models; however, it exhibited a vulnerability when the ACM's public key was replaced by a forged key, potentially compromising system security. The second improvement method, by incorporating digital certificates, effectively resolved the issues from the first method, with verification results indicating that all security properties were satisfied even under scenarios involving intruders. The innovative method eliminated dependence on the ACM, offering a simpler and more robust access control solution for small networks. The access control scheme updated by the innovative method from SVF-NDN is suitable to be applied in a small network, as it can decrease the complexity of messages between entities and avoid the situation that ACM is invaded. However, its efficiency will be reduced when dealing with a large network. Because the data transfer in NDN is based on the data name, there will be a lot of broadcast forwarding to transfer the *Interest* packet to the writer. Although users may encounter the risk of ACM being compromised, it is more appropriate to use the access control obtained by the improvement methods on a large network.

Methodology

We propose a Security Verification Framework for NDN Access Control (SFV-NDN) for the NDN access control scheme, which is proposed by Hamdane et al[8]. SFV-NDN integrates an approach to building, verifying, and enhancing access control mechanisms within network environments through CSP models. This process not only focuses on formalizing and verifying security properties but also adapts to evolving threat landscapes. The framework is illustrated in Fig. 1.

1. Build initial CSP models for access control: Initially, the access control requirements are captured in CSP models that define the system's behavior. This involves modeling various roles, such as readers, writers, and intruders, and their interactions. The inclusion of persistent intruder scenarios is emphasized in the model, eliminating the need for additional subscripts (e.g., subscript *I* used in prior work). This simplification makes the models more general while still addressing potential attack vectors.
2. Define properties by assertion: Critical security properties, such as deadlock freedom, data availability, key authentication, data leakage prevention, and data access protection, are formalized as assertions. These properties represent the essential security requirements of the system. They are expressed in a form suitable for formal verification using PAT, which can support LTL formulas.
3. Verify models by properties: Once the CSP model and properties are defined, the verification process begins. The models are fed into the PAT tool to check whether the system satisfies the specified security properties. SFV-NDN can effectively verify the properties users are concerned about. If all properties are satisfied, the process ends; otherwise, further analysis is needed.

4(a). Obtain the counter example if the verification result is invalid: If the security property is not satisfied, the verification tool provides a counter-example. This counter-example highlights specific flaws in the model, helping locate the issues that prevent the system from fulfilling its security requirements.
4(b). Return result if the verification result is invalid: If the properties are satisfied, the verification result is returned, confirming that the current model meets the security requirements.

4. Update CSP models using designed strategies by Algorithms 1–4: If the verification process uncovers issues, the CSP models are updated using a variety of strategies. SFV-NDN supports model updates based on analysis results, employing methods to enhance the system. Algorithms 1–4 provide an automated approach to refining the model, with multiple iterations of verification ensuring the system's robustness against attacks.
5. Update access control scheme: After updating the CSP models, the access control scheme itself is revised to reflect these improvements. The updated scheme is designed to ensure that critical security properties are maintained. By continuously refining the models and control schemes, the framework enhances system resilience against potential intrusions.This framework allows for iterative improvements by combining formal verification techniques with model adaptation strategies. The primary objective of SVF-NDN is to verify security properties and identify potential vulnerabilities. To achieve this, our framework focuses on five key security properties: deadlock freedom, data availability, key authentication, data leakage prevention, and data access protection. These properties are formalized as LTL formulas and verified using the PAT.

The input to SVF-NDN consists of NDN access control schemes, modeled using CSP. These models are then verified against the specified security properties. If any property fails the verification, the PAT tool generates a counterexample, pinpointing the exact vulnerabilities in the model. The output includes either verification results or counterexamples, which help identify where the system fails to meet security requirements.

To address these vulnerabilities, we have developed four algorithms that automate the process of updating the CSP models. These algorithms iteratively refine the models based on the verification results, thereby strengthening the system against attacks. By focusing on persistent attack scenarios and incorporating multiple verification cycles, SVF-NDN ensures that access control mechanisms remain robust and that predefined security properties are protected. The relationships between different CSP models are illustrated in Fig. 2, and further details on specific model scenarios will be discussed in subsequent sections.

Modeling entities in access control

In this section, the basic definitions of modeling are presented. The original CSP models which describe the access control formally are given. We will model entities and establish concurrent connections between entities.

Basic definitions of modeling

To model the behavior between the writers/readers and the ACM in Fig. 3, the fundamental information about sets, messages, and channels is needed.

The existence of six sets is assumed to be used in the models. **Entity** set represents entities including writers, readers, and ACM. **Name** set denotes NK names, Data names, and DK names. **Key** set is constituted by keys. **NKey** set is a subset of **Key** set including node key pairs and single node keys. **Content** set contains the content to be encrypted. **Ack** set consists of acknowledgments.The symbols and their meanings in the messages transmitted between entities, as well as their relationships with pre-defined sets, are illustrated in Table 1 and Table 2.

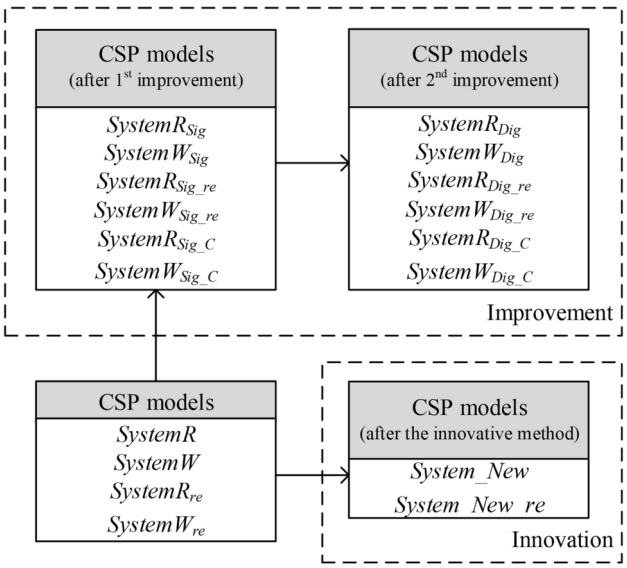

Fig. 2. Relationship between different CSP models.

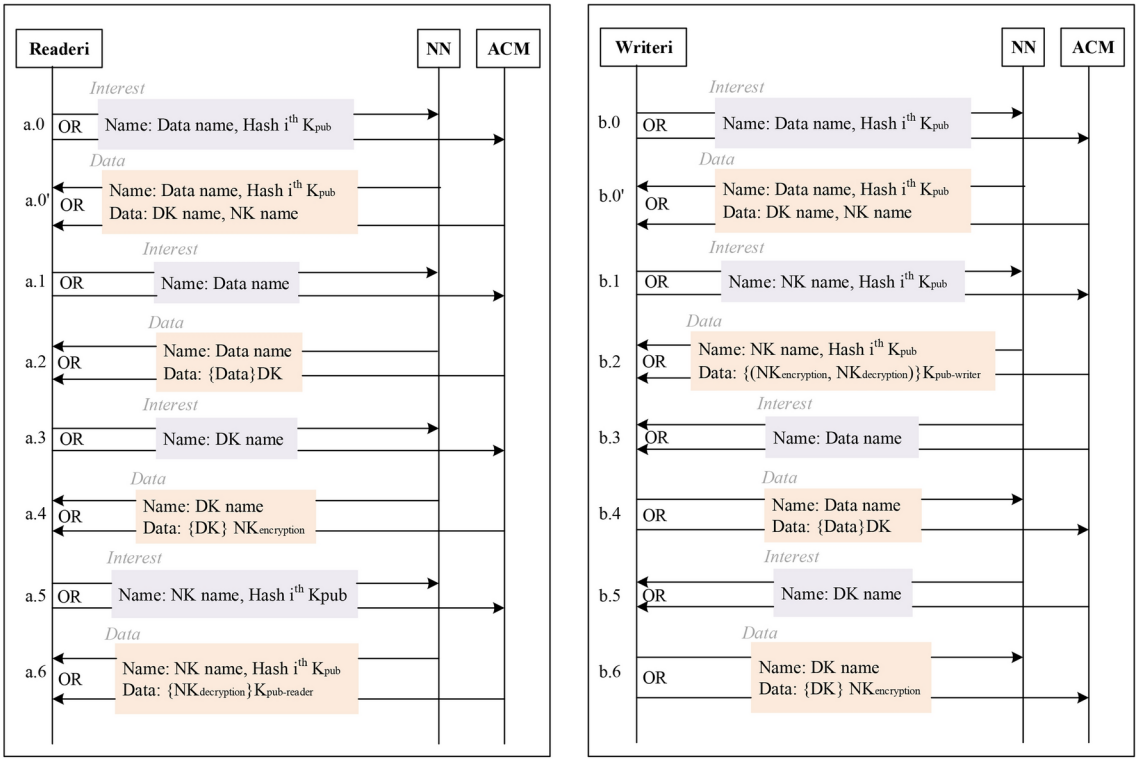

Fig. 3. Simplified NDN access control with read and write operations.

Set	Constants
Entity	R,R'(reader), W,W'(writer), M(ACM)
Name	NKN(node key pair name), DKN(data key name), DN(data name)
Key	K(public key), K^{-1}(private key), DK(data key), NK_e($NK_{encryption}$), NK_d($NK_{decryption}$), NK_e_f(fake $NK_{encryption}$), NK_d_f(fake $NK_{decryption}$), K_I(public key of intruder), K_M(public key of ACM), K_I^{-1}(private key of intruder), K_M^{-1}(private key of ACM), K_A(public key of CA), K_A^{-1}(private key of CA)
NKey	NK_e($NK_{encryption}$), NK_d($NK_{decryption}$), NK_e_f(fake $NK_{encryption}$), NK_d_f(fake $NK_{decryption}$)
Content	DATA(data)
Ack	YES(positive feedback), NO(negative feedback), Reject(reject feedback)

Table 1. The relationship between involved constants and pre-defined sets.

Set	Variables
Entity	r(reader), w(writer), m(ACM), e(entity)
Name	nkn(node key pair name), dkn(data key name), dn(data name)
Key	k,k',k1(public key), k^{-1},k'$^{-1}$,k1^{-1}(private key), dk,dk'(data key), nk_e,nk_e'(true/fake $NK_{encryption}$), nk_d(true/fake $NK_{decryption}$), $k_{m'}$,k_{m1}(public key of ACM), k_m^{-1},$k_{m'}^{-1}$,k_{m1}^{-1}(private key of ACM), k_a,$k_{a'}$(public key of CA), k_a^{-1},$k_{a'}^{-1}$(private key of CA)
NKey	nk_e,nk_e'(true/fake $NK_{encryption}$), nk_d(true/fake $NK_{decryption}$)
Content	data(data)
Ack	ack,ack'(positive/negative feedback), reject(reject feedback)

Table 2. The relationship between involved variables and pre-defined sets.

Model	Property				
	Deadlock Freedom	Data Availability	Key Authentication	Data Leakage Prevention	Data Access Protection
SystemR	*Valid*	*Valid*	*Not Valid*	*Valid*	*Valid*
SystemW	*Valid*	*Valid*	*Not Valid*	*Not Valid*	*Valid*
SystemR	*Valid*	*Not Valid*	*Valid*	*Valid*	*Valid*
SystemW	*Valid*	*Not Valid*	*Valid*	*Valid*	*Valid*
SystemR_{Sig}	*Valid*	*Valid*	*Valid*	*Valid*	*Valid*
SystemW_{Sig}	*Valid*	*Valid*	*Valid*	*Valid*	*Valid*
SystemR_{Sig_re}	*Valid*	*Not Valid*	*Valid*	*Valid*	*Valid*
SystemW_{Sig_re}	*Valid*	*Not Valid*	*Valid*	*Valid*	*Valid*
SystemR_{Sig_C}	*Valid*	*Valid*	*Not Valid*	*Valid*	*Valid*
SystemW_{Sig_C}	*Valid*	*Valid*	*Not Valid*	*Not Valid*	*Valid*
SystemR_{Dig}	*Valid*	*Valid*	*Valid*	*Valid*	*Valid*
SystemW_{Dig}	*Valid*	*Valid*	*Valid*	*Valid*	*Valid*
SystemR_{Dig_re}	*Valid*	*Not Valid*	*Valid*	*Valid*	*Valid*
SystemW_{Dig_re}	*Valid*	*Not Valid*	*Valid*	*Valid*	*Valid*
SystemR_{Dig_C}	*Valid*	*Valid*	*Valid*	*Valid*	*Valid*
SystemW_{Dig_C}	*Valid*	*Valid*	*Valid*	*Valid*	*Valid*

Table 3. Verification Results of Models.

Model	Property			
	Deadlock Freedom	Data Availability	Data Leakage Protection	Data Access Protection
System_New	Valid	Valid	Valid	Valid
System_New_re	Valid	Not Valid	Valid	Valid

Table 4. Verification Results of Innovative Method.

Interest and *Data* packets transmitted between entities and internal processing procedures of entities are two core elements of modeling. With the help of the previously defined sets, they are abstracted into different messages. Each message is tagged with one of the labels from the set $\{\text{msg}_{\text{int}}, \text{msg}_{\text{dat}}, \text{msg}_{\text{ack}}, \text{msg}_{\text{pro}}\}$. These labels indicate the type of the current message, where msg_{int} corresponds to *Interest* packets, msg_{dat} corresponds to *Data* packets, msg_{ack} represents acknowledgment, and msg_{pro} denotes internal processing. In addition, the form $E(k)$ and $H(k)$ are used to represent the encryption and the hash of key k. At the same time, the keywords Name and Data are used in the messages to better distinguish between *Interest* packets and *Data* packets. Here four Boolean variables considering verification are defined. They are *data_acquisition_success*, *nk_faking_success*, *data_leakage_success* and *is_rejected*, which are initialized to false. When their values are true, these mean that data is obtained by the reader or ACM, the node key pair is faked, data is obtained by intruders and the revoked reader or writer is rejected by ACM respectively.

Reader modeling

We construct the processes that characterize the read and write operations, namely, processes *SystemR* and *SystemW*. Fig. 4 illustrates the interprocess communication between processes in *SystemR* and *SystemW*, representing the read and write operations with the participation of an intruder, respectively. Entities are represented by rectangles, where *READER* and *WRITER*, as indicated by their names, represent the reader and the writer, respectively. *ACM_R* and *ACM_W* symbolize the behavior of ACM when communicating with readers and writers. *PROCESS* denotes the internal processing procedure. In the presence of intruders, the *INTRUDER* process is created to simulate the behavior of intruders who eavesdrop on and modify messages. The arrowed connections between entities represent different channels.

Additionally, $\text{SystemR}_{\text{re}}$ is akin to *SystemR* but involves a revoked reader. $\text{SystemW}_{\text{re}}$ employs a revoked writer, with the remaining part identical to *SystemW*. Due to space limitations, modeling for readers is provided. An algorithm is designed to facilitate the generation of the models. Process READER_0 is formalized to describe the behavior of a reader.

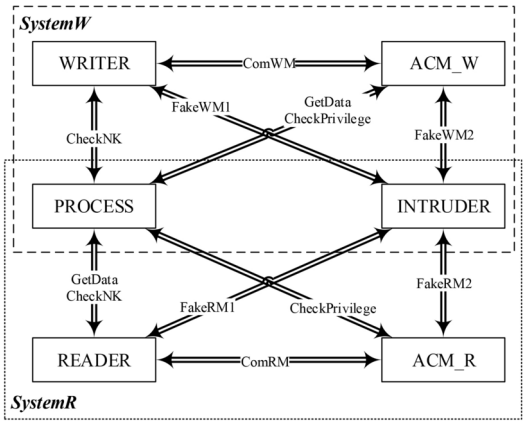

Fig. 4. Interprocess communication between processes in CSP models.

$$
\begin{aligned}
&\mathrm{READER}_0(\mathrm{r,m,k,dn}) =_{df} \ \mathrm{ComRM!msg_{int}.r.m.}\mathit{Name}(\mathrm{dn,H(k)})\\
&\quad \rightarrow \begin{pmatrix} (\mathrm{ComRM?msg_{pro}.reject} \rightarrow \mathrm{SKIP}) \\ \square \mathrm{READER}_0\_\mathrm{Sub(r,m,k,dn)} \end{pmatrix}; \mathrm{READER}_0(\mathrm{r,m,k,dn})\\
&\mathrm{READER}_0\_\mathrm{Sub(r,m,k,dn)} =_{df} \ \mathrm{ComRM?msg_{dat}.m.r.}\mathit{Data}(\mathrm{dkn,nkn})\\
&\quad \rightarrow \mathrm{ComRM!msg_{int}.r.m.}\mathit{Name}(\mathrm{dn}) \rightarrow \mathrm{ComRM?msg_{dat}.m.r.}\mathit{Data}(\mathrm{E(dk,data)}) \rightarrow\\
&\quad \mathrm{ComRM!msg_{int}.r.m.}\mathit{Name}(\mathrm{dkn}) \rightarrow \mathrm{ComRM?msg_{dat}.m.r.}\mathit{Data}(\mathrm{E(nk\_e,dk')})\\
&\quad \rightarrow \mathrm{ComRM!msg_{int}.r.m.}\mathit{Name}(\mathrm{nkn,H(k)}) \rightarrow\\
&\quad \mathrm{ComRM?msg_{dat}.m.r.}\mathit{Data}(\mathrm{E(k',nk\_d)}) \rightarrow \mathrm{CheckNK!msg_{pro}.E(k',nk\_d).k^{-1}} \rightarrow \mathrm{CheckNK?msg_{ack}.ack} \rightarrow\\
&\quad \begin{pmatrix} (\mathrm{NKFakingSuccess\{nk\_faking\_success = true\}} \rightarrow \mathrm{SKIP}) \\ \lhd (\mathrm{ack == YES}) \rhd (\mathrm{NKFakingError\{nk\_faking\_success = false\}} \rightarrow \mathrm{SKIP}) \end{pmatrix};\\
&\quad \mathrm{GetData!msg_{pro}.E(dk,data).E(nk\_e,dk').nk\_d} \rightarrow\\
&\quad \mathrm{GetData?msg_{ack}.ack'} \rightarrow \begin{pmatrix} (\mathrm{DataAcquisitionSuccess\{data\_acquisition\_success = true\}} \rightarrow \mathrm{SKIP}) \\ \lhd (\mathrm{ack' == YES}) \rhd (\mathrm{DataAcquisitionSuccess\{data\_acquisition\_success} \\ \mathrm{= false\}} \rightarrow \mathrm{SKIP}) \end{pmatrix}
\end{aligned}
$$

When the reader's reading privilege is revoked, the reader will receive a reject message from ACM. If it owns reading privilege, then the corresponding read operations are performed. The first eight actions on channel *ComRM* correspond to steps a.0, a0'and a.1 - a.6 of Reader_i in Fig.3 in order. By channel *CheckNK*, whether the value carried by *nk_d* is faked or not can be checked. Whether *data* is obtained successfully using channel *GetData* can be learned.

In order to permit the possibility of intruder action, the message on the *ComRM* channel is allowed to be intercepted or faked. This can be achieved through a process called renaming. Moreover, $\{|c|\}$ represents the set of all communications over channel *c*. Algorithm 1 is designed to support this renaming process. Furthermore, by using this algorithm, the following process *READER* can be obtained:

$$
\begin{aligned}
&\mathrm{READER(r,m,k,dn)} =_{df}\\
&\qquad \mathrm{READER}_0\mathrm{(r,m,k,dn)}[[\mathrm{ComRM?\{|ComRM|\}} \leftarrow \mathrm{ComRM?\{|ComRM|\}},\\
&\qquad \mathrm{ComRM?\{|ComRM|\}} \leftarrow \mathrm{FakeRM1?\{|ComRM|\}}, \mathrm{ComRM!\{|ComRM|\}}\\
&\qquad \leftarrow \mathrm{ComRM!\{|ComRM|\}}, \mathrm{ComRM!\{|ComRM|\}} \leftarrow \mathrm{FakeRM1!\{|ComRM|\}}]]
\end{aligned}
$$

READER will perform either an action on channel *ComRM* or channel *FakeRM1* whenever READER_0 performs a corresponding action on channel *ComRM*. Besides, *READER* and READER_0 perform the same action. The purpose of Algorithm 1 is to support channel renaming in the CSP model of an entity. By using this algorithm, the CSP definition of an entity can be mapped from one channel to another, enabling channel adjustments in different communication contexts. For example, in the instance shown in Fig. 5, Algorithm 1 is applied to rename channels between READER_0 and *READER*. Specifically, *ComRM* channel is renamed to *FakeRM1*, achieved through channel substitution in the CSP process definition. In this scenario, Algorithm 1 ensures that both channels remain usable, thereby maintaining system flexibility and functionality.

The first improvement method for the access control

To maintain key authentication and prevent data leakage in access control during cyberattacks, an approach similar to digital signature is proposed. The modifications in constructing novel CSP models while implementing

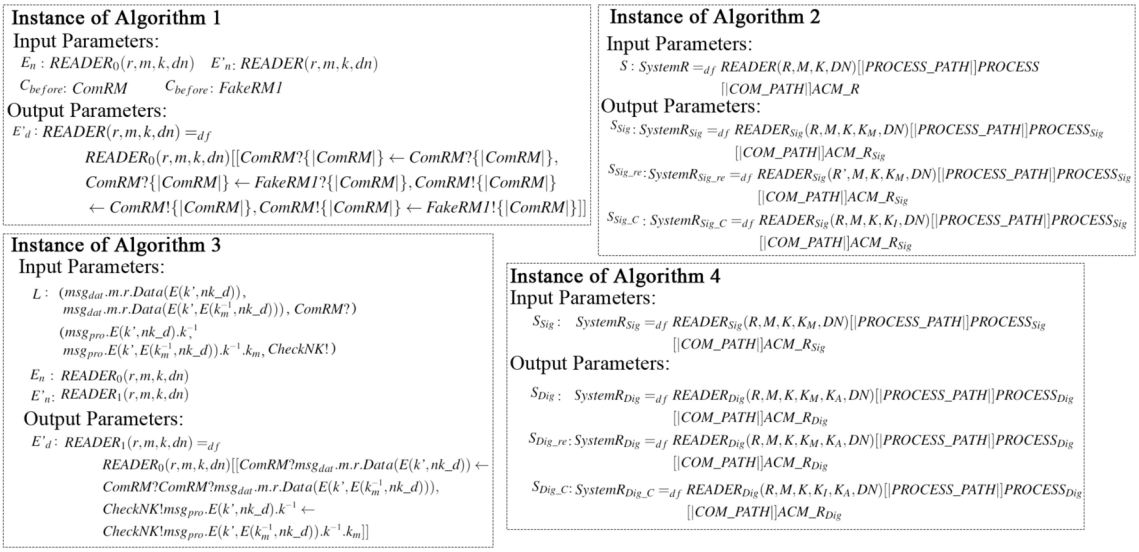

Fig. 5. Instance of Algorithms 1, 2, 3 and 4.

this technique are explained. Due to space constraints, only the modeling for the reader and writer are provided. Algorithm 1 and Algorithm 2 are introduced for the facile creation of models. Lastly, the effectiveness of the novel CSP models is validated through a thorough verification process.

Overall modeling

In the first improvement method, the ACM creates a special pair of keys: public key K_M and private key K_M^{-1}. Assuming that K_M is known by readers and writers. When producing message a.6 and message b.2 in Fig. 3, ACM uses its private key K_M^{-1} to encrypt ($\mathrm{NK_{encryption}}$,$\mathrm{NK_{decryption}}$) and $\mathrm{NK_{decryption}}$ at first, like creating a digital signature. This happens before ($\mathrm{NK_{encryption}}$,$\mathrm{NK_{decryption}}$) and $\mathrm{NK_{decryption}}$ are encrypted by the public key of readers and writers. When readers and writers receive these messages, they apply K_M to decrypt the digital signature in them. If the operation is successful, it can be inferred that the message has been sent by the ACM.

Input: CSP process name E_n of entity E with parameters before renaming.
CSP process name E'_n of entity E' with parameters after renaming.
Channel name C_{before} before renaming.
Channel name C_{after} after renaming.
Output: a string of the definition E'_d of entity E'.

```
string a ← C_before + "?{|" + C_before + "|}"
string b ← C_before + "!{|" + C_before + "|}"
string c ← C_after + "?{|" + C_before + "|}"
string d ← C_after + "!{|" + C_before + "|}"
E'_d ← E'_n + "=_df" + E_n + "[[" +
        a + "←" + a + "," + a + "←" + c + "," +
        b + "←" + b + "," + b + "←" + d + "]]"
return E'_d
```

Algorithm 1. Renaming channels in entity's CSP model.

The read and write operations are re-modeled with the appearance of intruders. Updating *SystemR* and *SystemW* results in the generation of six updated CSP models in the first improvement. $\mathrm{SystemR_{Sig}}$ and $\mathrm{SystemW_{Sig}}$ represent the situation of applying the digital signature. $\mathrm{SystemR_{Sig\_re}}$ and $\mathrm{SystemW_{Sig\_re}}$ are the cases of using revoked entities. Due to space constraints, only the definitions of $\mathrm{SystemR_{Sig}}$ and $\mathrm{SystemW_{Sig}}$ are shown here. Algorithm 2 is designed to update CSP models for improvement, producing three updated models: $\mathrm{S_{Sig}}$, $\mathrm{S_{Sig\_re}}$ and $\mathrm{S_{Sig\_C}}$. In Fig. 5 for instance, the initial model *S* is defined as a combination of *READER*, *PROCESS*, and *ACM_R* processes connected by communication paths. After adding

the *Sig* subscript and introducing K_M, the updated S_{Sig} is formed. Next, modifying *R* to *R*' results in S_{Sig_re}, representing a state change. Finally, replacing K_M with K_I generates S_{Sig_C}, indicating an adjustment in the authentication parameter. These transformations illustrate how the model structure remains consistent while enabling functional extensions through parameter changes.

Input: the definition of CSP models *S*.
Output: the definitions of CSP models the in the first improvement, which are S_{Sig}, S_{Sig_re} *and* S_{Sig_C}.
1: S_{Sig} ← *Add subscript Sig to the name of processes, and add* K_M *to the parameters of entites' processes in S's definition.*
2: S_{Sig_re} ← *Modify the first parameter in* S_{Sig}*'s definition to include a prime mark (').*
3: S_{Sig_C} ← *Replace the* K_M *in the parameters of processes in* S_{Sig}*'s definition with* K_I.
4: ***return*** S_{Sig}, S_{Sig_re} *and* S_{Sig_C}

Algorithm 2. Updating CSP overall models in the first improvement.

Reader and writer modeling
Fig. 6 depicts the modeling update for the reader. Specifically, READER_1 is derived from READER_0 through renaming, resulting in a structure similar to READER_0. Algorithm 3 facilitates the updating of READER_0 to READER_1, supporting the modification of actions in an entity's CSP model. Algorithm 3 is designed to update actions within a CSP model by transforming a process E_n into an updated version E'_n. The algorithm takes a list *L* containing triples of the original message, the updated message, and the associated communication channel. Using this information, it iteratively constructs the new process E'_d, where each specified action is updated on its respective channel. The process ensures proper formatting and outputs the updated definition as a string. For example, in the instance shown in Fig. 5, the original process $\mathrm{READER}_0(r,m,k,dn)$ is updated to $\mathrm{READER}_1(r,m,k,dn)$. The updates include replacing $\mathrm{msg_{dat}}.m.r.\mathrm{Data}(E(k',nk\_d))$ with $\mathrm{msg_{dat}}.m.r.\mathrm{Data}(E(k', E(\mathrm{k_m^{-1}},nk\_d)))$ on the channel *ComRM*, and $\mathrm{msg_{pro}}.E(k',nk\_d).\mathrm{k}^{-1}$ with $\mathrm{msg_{pro}}.E(k',E(\mathrm{k_m^{-1}},nk\_d)).\mathrm{k}^{-1}.\mathrm{k_m}$ on the channel *CheckNK*. The resulting definition of $READER_1$ integrates these changes, reflecting the updated actions on the specified channels, demonstrating the algorithm's capability to manage precise modifications in CSP processes.

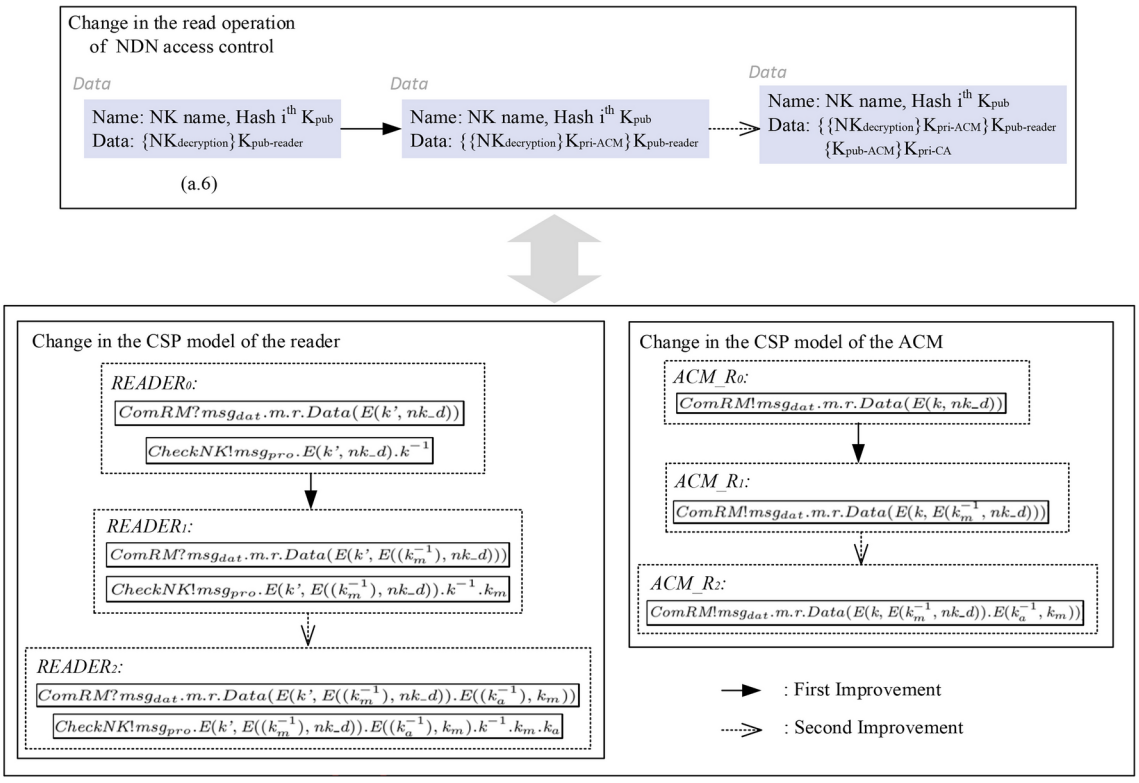

Fig. 6. Corresponding changes in the NDN access control and CSP models for the read operation.

The only distinction lies in the fact that READER_1 updates two actions on channel *ComRM* and channel *CheckNK* respectively. Simultaneously, in order to update the transmitted packet (a.6) in Fig. 6, the private key of ACM is employed to encrypt the node key first, followed by encryption using the reader's public key. A similar modeling approach can be employed to derive $\mathrm{ACM\_R}_1$ based on ACM_R_0, as illustrated in Fig. 6.

Input: a list L consists of triples (messages before and after modification, and the common channel with ! or ?).
CSP process name E_n of entity E with parameters.
CSP process name E'_n of entity E' with parameters after the update on E.
Output: a string of the definition E'_d of entity E'.

$n_L \leftarrow$ the length of L
$E'_d \leftarrow E'_n$ + "$=_{df}$" + E_n + "[["
for $i \leftarrow 0$ to $n_L - 1$ **do**
  string triple $(msg_{before}, msg_{after}, chn) \leftarrow L[i]$
  $E'_d \leftarrow E'_d + chn + msg_{before}$ + "←" + $chn + msg_{after}$
  if $i = n_L - 1$ **then**
    $E'_d \leftarrow E'_d$ + "]]"
  else
    $E'_d \leftarrow E'_d$ + ","
  end if
end for
return E'_d

Algorithm 3. Updating actions in entity's CSP model.

Meanwhile, renaming is performed to simulate interception and forgery on channel *ComRM*. Consequently, $\mathrm{READER}_{\mathrm{Sig}}$ can be generated by using *READER* as input for Algorithm 1. Whenever either an action on channel *ComRM* or channel *FakeRM1* is done in READER_1, a corresponding action is done on channel *ComRM* in $\mathrm{READER}_{\mathrm{Sig}}$. Except that, $\mathrm{READER}_{\mathrm{Sig}}$ and READER_1 perform the same action.

To update the transmitted packet (b.2) in Fig. 7, the process involves using the private key of ACM to encrypt the node key pair, followed by encryption using the writer's public key. Employing a modeling approach similar to that of the reader, corresponding entities WRITER_1 and ACM_W_1 can be obtained, as depicted in Fig. 7.

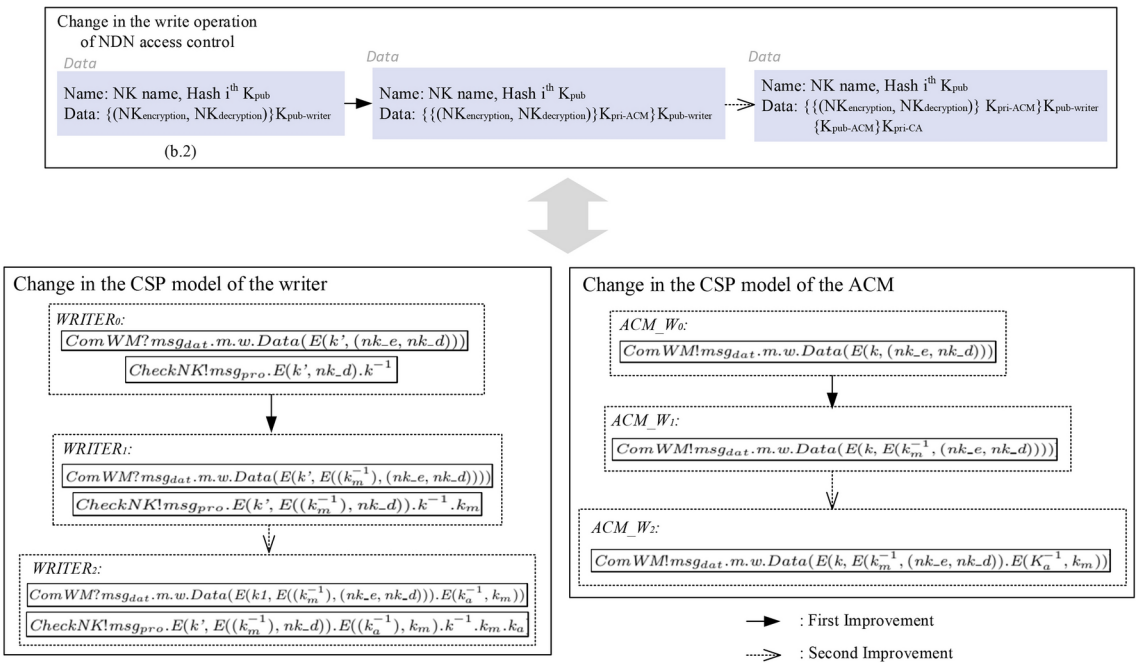

Fig. 7. Corresponding changes in the NDN access control and CSP models for the write operation.

The second improvement method for access control

According to the previous analysis, the first improvement method does effectively protect keys and data, but it is still weak in handling special cases when ACM's public key known by the entity is faked by an intruder's public key. Therefore, a second improvement is proposed to address this problem. This section focuses on presenting the updates made during the modeling process and verifying the security properties of the access control under the new solution. Meanwhile, Algorithm 4 is also given to support better model building.

Overall modeling

The second improvement is to update the previous method by adding digital certificates. Specifically, ACM provides its public key $\mathrm{K_M}$ to the certificate authority (CA). CA uses its private key K_A^{-1} to encrypt K_M, which creates a digital certificate $E((\mathrm{K}_A^{-1}),\mathrm{K}_M)$. After performing the same operations as those in the first model improvement, the digital certificate is added to the end of message a.6 and message b.2. Assuming that readers and writers know CA's public key K_A, they will fetch the digital certificates at first when dealing with these two messages. Then they decrypt the digital certificates using K_A to get $\mathrm{K_M}$, which will be compared with the public key of the ACM known by the reader/writer. If the two keys match, it can be concluded that the public key known by the reader/writer is the authentic key of the ACM. Otherwise, the key is deemed to be forged.

Except for adding CA's public key $\mathrm{K_A}$ as a parameter, the six models here are almost having the same definitions as the six models in the first improvement method. $\mathrm{SystemR_{Dig}}$ and $\mathrm{SystemW_{Dig}}$ represent the situation of using the digital certificates. $\mathrm{SystemR_{Dig\_re}}$ and $\mathrm{SystemW_{Dig\_re}}$ indicate the cases of having revoked entities. $\mathrm{SystemR_{Dig\_C}}$ and $\mathrm{SystemW_{Dig\_C}}$ denote the public key of ACM known by the reader and writer are replaced by the public key of the intruder. They can be obtained conveniently by applying Algorithm 4.

Algorithm 4 focuses on the second improvement of CSP models, transforming $\mathrm{S_{Sig}}$ into three updated models: $\mathrm{S_{Dig}}$, $\mathrm{S_{Dig\_re}}$ and $\mathrm{S_{Dig\_C}}$. For example, in the instance shown in Fig. 5, Algorithm 4 generates S_{Dig} by renaming the processes and adding K_A to the parameters. The updated model S_{Dig_re} modifies R to R', while S_{Dig_C} replaces K_M with K_I, reflecting a change in the cryptographic configuration. This step-by-step transformation illustrates the systematic enhancement of CSP models for more robust applications.

Input: the definition of CSP models S_{Sig}.
Output: the six definitions of CSP models the in the second improvement, which are S_{Dig}, S_{Dig_re} and S_{Dig_C}.
1: S_{Dig} ← Change subscript *Sig* on the name of processes to *Dig*, and add K_A to the parameters of entites' processes in S_{Sig}'s definition.
2: S_{Dig_re} ← Modify the first parameter in S_{Dig}'s definition to include a prime mark (').
3: S_{Dig_C} ← Replace the K_M in the parameters of processes in S_{Dig}'s definition with K_I.
4: **return** S_{Dig}, S_{Dig_re} and S_{Dig_C}

Algorithm 4. Updating CSP overall models in the second improvement.

Reader and writer modeling

To handle the transmitted packet (a.6) in Fig. 6, the second improvement involves adding a digital certificate to the packet. Consequently, $\mathrm{READER_2}$ is created through renaming from $\mathrm{READER_1}$, shown in Fig. 6. Similarly, Algorithm 3 supports the construction of $\mathrm{READER_2}$ based on $\mathrm{READER_1}$. To simulate interception and forgery on channel *ComRM*, renaming is applied.

Utilizing Algorithm 1, it is possible to construct $\mathrm{READER_{Dig}}$ based on $\mathrm{READER_2}$. Whenever $\mathrm{READER_2}$ executes an action on channel *ComRM*, *READER_Dig* will perform either a corresponding action on channel *ComRM* or on channel *FakeRM1*. In addition to that, *READER_Dig* carries out the same action as $\mathrm{READER_2}$.

To manage the transmitted packet (b.2) in Fig. 7, the second improvement entails incorporating a digital certificate into the packet. By employing a modeling approach similar to that of the reader, corresponding entities $\mathrm{WRITER_2}$ and $\mathrm{ACM\_W_2}$ can be obtained, as depicted in Fig. 7.

The initial CSP model and all the CSP models involved in the improvement methods have been implemented in PAT. Please refer to the website(The full implementation of the improvement methods in PAT can be found at https://github.com/asunafy/SVF-NDN.) for full implementation.

The innovative method for the access control

The improvement method for NDN access control has been adopted, demonstrating promising performance in enhancing security when the ACM is secure. When the ACM is unsafe, the improvement method does not perform well. Therefore, in this section, a new access control method is introduced that does not rely on considering the ACM, presenting an innovative approach to access control. SFV-NDN is also applied to accomplish the update of the CSP models and the access control. The verification results of this method are illustrated and analyzed.

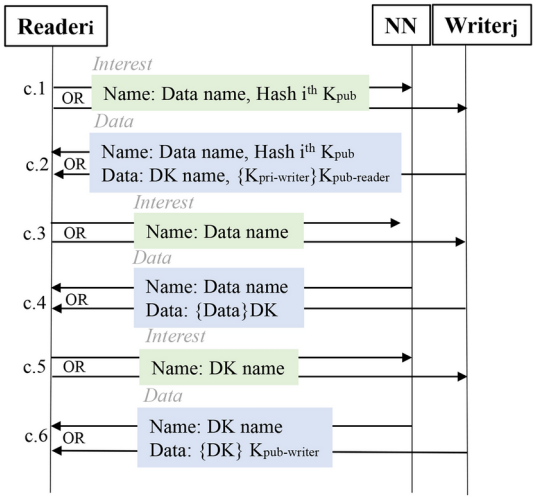

Fig. 8. New NDN access control with read and write operations.

Set	Constants	Variables
Entity'	R(reader), W(writer)	r(reader), w(writer)
Name'	DKN(data key name), DN(data name)	dkn(data key name), dn(data name)
Key'	K(public key), K^{-1}(private key), DK(data key), K_I(public key of intruder), K_I^{-1}(private key of intruder)	k,k',k1,k2(public key), dk,dk'(data key), k^{-1},k'^{-1},$k1^{-1}$,$k2^{-1}$ (private key)
PriKey	K^{-1}(private key), K_I^{-1}(private key of intruder)	k^{-1},k'^{-1},$k1^{-1}$,$k2^{-1}$ (private key)

Table 5. The new pre-defined sets in the new access control.

Overall modeling

The new access control is shown in Fig.8. If a reader wants to subscribe to the data from a writer, he first applies special commands to request the naming convention and checks if he has the read privilege of the data (step c.1). Then the reader receives the name of the data key DK and the private key of the writer encrypted by his public key (step c.2). After receiving the data request (step c.3), the writer produces the data by encrypting its private key by the public key of the reader (step c.4). The reader requests for the data key (step c.5). The writer returns the data key *DK* encrypted by his public key (step c.6).

To specify the new access control, several new sets are defined. **Entity**' set denotes writers and readers. **Name**' set represents data names and data key names. **Key**' set contains keys in the new access control. As a subset of **Key**', **PriKey** set consists of private keys. The new access control also uses **Content** set and **Ack** set from the old access control. The elements of the new pre-defined sets are listed in Table 5.

The tag set used in the old access control is inherited, and the messages are defined as follows.

$$\mathrm{MSG}'_{\mathrm{int}} = \{\mathrm{msg}_{\mathrm{int}}.\mathrm{a.b}.Name(\mathrm{n}), \mathrm{msg}_{\mathrm{int}}.\mathrm{a.b}.Name(\mathrm{n},\mathrm{H}(\mathrm{k})) \mid \mathrm{a},\mathrm{b} \in \mathrm{Entity}', \mathrm{n} \in \mathrm{Name}', \mathrm{k} \in \mathrm{Key}'\}$$
$$\mathrm{MSG}'_{\mathrm{dat1}} = \{\mathrm{msg}_{\mathrm{dat}}.\mathrm{a.b}.Data(\mathrm{E}(\mathrm{k},\mathrm{c})) \mid \mathrm{a},\mathrm{b} \in \mathrm{Entity}', \mathrm{k} \in \mathrm{Key}', \mathrm{c} \in (\mathrm{Key}' - \mathrm{PriKey}) \cup \mathrm{Content}\}$$
$$\mathrm{MSG}'_{\mathrm{dat2}} = \{\mathrm{msg}_{\mathrm{dat}}.\mathrm{a.b}.Data(\mathrm{E}(\mathrm{k},\mathrm{c})) \mid \mathrm{a},\mathrm{b} \in \mathrm{Entity}', \mathrm{n} \in \mathrm{Name}', \mathrm{k} \in \mathrm{Key}', \mathrm{c} \in \mathrm{PriKey}\}$$
$$\mathrm{MSG}'_{\mathrm{ack}} = \{\mathrm{msg}_{\mathrm{ack}}.\mathrm{x} \mid \mathrm{x} \in \mathrm{Ack}\}$$
$$\mathrm{MSG}'_{\mathrm{pro}} = \{\mathrm{msg}_{\mathrm{pro}}.\mathrm{E}(\mathrm{k}_1,\mathrm{k}_2).\mathrm{E}(\mathrm{k}_3,\mathrm{c}).\mathrm{E}(\mathrm{k}_4,\mathrm{k}_5).\mathrm{k}_6 \mid \mathrm{k}_1,\mathrm{k}_2,\mathrm{k}_3,\mathrm{k}_4,\mathrm{k}_5,\mathrm{k}_6 \in \mathrm{Key}', \mathrm{c} \in \mathrm{Key}' \cup \mathrm{Content}\}$$
$$\mathrm{MSG}'_{\mathrm{out}} = \mathrm{MSG}'_{\mathrm{int}} \cup \mathrm{MSG}'_{\mathrm{dat1 \cup dat2}} \qquad \mathrm{MSG}'_{\mathrm{in}} = \mathrm{MSG}'_{ack} \cup \mathrm{MSG}'_{\mathrm{pro}}$$
$$\mathrm{MSG}' = \mathrm{MSG}'_{\mathrm{in}} \cup \mathrm{MSG}'_{\mathrm{out}}$$

The declarations of channels are as below.

$$\mathrm{Channel\ ComRW, FakeRW1, FakeRW2} : \mathrm{MSG}'_{\mathrm{out}}$$
$$\mathrm{Channel\ GetData} : \mathrm{MSG}'_{\mathrm{in}}$$

The entire models are composed of the following modules as below.

$$
\begin{aligned}
\mathrm{System\_New} =_{df}\ & \mathrm{READER\_New(R, W, K, ND)[|GetData'|]PROCESS\_New} \\
& [|\mathrm{ComRW}|]\mathrm{WRITER\_New(W, R, NDK, ND, DK, DATA)} \\
& [|\mathrm{FakeRW1, FakeRW2}|]\mathrm{INTRUDER\_New} \\
\mathrm{System\_New\_re} =_{df}\ & \mathrm{READER\_New(R', W, K, ND)[|GetData'|]PROCESS\_New} \\
& [|\mathrm{ComRW}|]\mathrm{WRITER\_New(W, R', NDK, ND, DK, DATA)} \\
& [|\mathrm{FakeRW1, FakeRW2}|]\mathrm{INTRUDER\_New}
\end{aligned}
$$

READER_New, *WRITER_New* and *PROCESS_New* represent the reader, writer, and the processing procedure in the new access control. *INTRUDER_New* stands for the intruder, especially for the new access control.

Reader and writer modeling

READER' and *WRITER*' simulate the operations of readers and writers in the new access control. There are six actions on channel *ComRW* corresponding to step c.1 - step c.6 in Fig.8. By utilizing the *GetData* channel, it can be determined whether the *data* is obtained successfully. *WRITER*' also has six actions on channel *ComRW* which are related to step c.1 - step c.6 in Fig.8.

The definition of *READER*' and *WRITER*' are as below.

$$
\begin{aligned}
& \mathrm{READER'(r, w, k, dn)} =_{df} \\
& \quad \mathrm{ComRW!msg_{int}.r.w.}Name(\mathrm{dn, H(k)}) \rightarrow \begin{pmatrix} (\mathrm{ComRW?msg_{pro}.reject} \rightarrow \mathrm{SKIP}) \\ \Box \mathrm{READER'\_Sub(r, w, k, dn)} \end{pmatrix}; \\
& \quad \mathrm{READER'(r, w, k, dn)} \\
& \mathrm{READER'\_Sub(r, w, k, dn)} =_{df} \\
& \quad \mathrm{ComRW?msg_{dat}.w.r.}Data(dkn, \mathrm{E(k', k1^{-1})}) \rightarrow \mathrm{ComRW!msg_{int}.r.w.}Name(\mathrm{dn}) \rightarrow \\
& \quad \mathrm{ComRW?msg_{dat}.w.r.}Data(\mathrm{E(dk, data)}) \rightarrow \mathrm{ComRW!msg_{int}.r.w.}Name(\mathrm{dkn}) \rightarrow \\
& \quad \mathrm{ComRW?msg_{dat}.w.r.}Data(\mathrm{E(k2, dk')}) \rightarrow \mathrm{GetData!msg_{pro}.E(k', k1^{-1}).E(dk, data).E(k2, dk').k^{-1}} \rightarrow \\
& \quad \mathrm{GetData?msg_{ack}.ack} \rightarrow \begin{pmatrix} (\mathrm{DataAcquisitionSuccess\{d = true\}} \rightarrow \mathrm{SKIP}) \\ \lhd (\mathrm{ack == YES}) \rhd (\mathrm{DataAcquisitionError\{d = false\}} \rightarrow \mathrm{SKIP}) \end{pmatrix} \\
& \mathrm{WRITER'(w, r, k, dn, dkn, data, dk)} =_{df} \\
& \quad \mathrm{ComRW?msg_{int}.r.w.}Name(\mathrm{dn, H(k')}) \rightarrow \mathrm{CheckPrivilege!msg_{pro}.r} \rightarrow \\
& \quad \mathrm{CheckPrivilege?msg_{ack}.ack} \rightarrow \\
& \quad \begin{pmatrix} (\mathrm{RevokedEntityDeniedSuccess\{is\_rejected = true\}} \rightarrow \mathrm{SKIP} \\ \lhd (\mathrm{ack == YES}) \rhd \\ (\mathrm{RevokedEntityDeniedSuccess\{is\_rejected = false\}} \rightarrow \\ \mathrm{WRITER'\_Sub(w, r, k, dn, dkn, data, dk)} \end{pmatrix}; \\
& \quad \mathrm{WRITER'(w, r, k, dn, dkn, data, dk)} \\
& \mathrm{WRITER'\_Sub(w, r, k, dn, dkn, data, dk)} =_{df} \\
& \quad \mathrm{ComRW!msg_{dat}.w.r.}Data(dkn, \mathrm{E(k', k^{-1})}) \rightarrow \mathrm{ComRW?msg_{int}.r.w.}Name(\mathrm{dn}) \rightarrow \\
& \quad \mathrm{ComRW!msg_{dat}.w.r.}Data(\mathrm{E(dk, data)}) \rightarrow \mathrm{ComRW?msg_{int}.r.w.}Name(\mathrm{dkn}) \rightarrow \\
& \quad \mathrm{ComRW!msg_{dat}.w.r.}Data(\mathrm{E(k, dk)}) \rightarrow \mathrm{SKIP}
\end{aligned}
$$

If the reader's reading right is revoked, the writer will reject the request. Otherwise, the corresponding operations are performed. For *READER*' and *WRITER*', the six actions on channel *ComRW* correspond to step c.1 - c.6 in Fig.8 in order. The six actions of *READER*' and *WRITER*' on channel *ComRW* correspond to each other. *Reader*' first sends the *Interest* packet including the data name and the public key hash value. Then *WRITER*' returns the *Data* packet with the data key name and its private key encrypted with the reader's public key. *READER*' also sends the data's name to *Writer*', and then gets the *Data* packet including the encrypted data. Finally, *READER*' transmits the DK's name to *Writer*', and *WRITER*' feeds back with the *Data* packet carrying encrypted data key.

To simulate the actions of intruders, renaming is employed to simulate the interception and forgery of messages on the *ComRW* channel. *READER_New* and *WRITER_New* are built from Algorithm 1 as below.

$$\begin{aligned}
&\mathrm{READER_New(r,w,k,nr,dn)} =_{df}\\
&\quad \mathrm{READER'(r,w,k,nr,dn)}\\
&\quad [[\mathrm{ComRW?\{|ComRW|\}} \leftarrow \mathrm{ComRW?\{|ComRW|\}}, \mathrm{ComRW?\{|ComRW|\}} \leftarrow \mathrm{FakeRW1?\{|ComRW|\}},\\
&\quad \mathrm{ComRW!\{|ComRW|\}} \leftarrow \mathrm{ComRW!\{|ComRW|\}}, \mathrm{ComRW!\{|ComRW|\}} \leftarrow \mathrm{FakeRW1!\{|ComRW|\}}]]\\
&\mathrm{WRITER_New(w,m,k,dn,dkn,data,dk)} =_{df}\\
&\quad \mathrm{WRITER'(w,m,k,dn,dkn,data,dk)}\\
&\quad [[\mathrm{ComRW?\{|ComRW|\}} \leftarrow \mathrm{ComRW?\{|ComRW|\}}, \mathrm{ComRW?\{|ComRW|\}} \leftarrow \mathrm{FakeRW2?\{|ComRW|\}},\\
&\quad \mathrm{ComRW!\{|ComRW|\}} \leftarrow \mathrm{ComRW!\{|ComRW|\}}, \mathrm{ComRW!\{|ComRW|\}} \leftarrow \mathrm{FakeRW2!\{|ComRW|\}}]]
\end{aligned}$$

Whenever either an action on channel *ComRW* or channel *FakeRW1* is done in *READER*', a corresponding action is done on channel *ComRW* in *READER_New*. Except that, *READER_New* and *READER*' perform the same action. *WRITER_New* performs the corresponding action which is done on channel *ComRW*, whenever *WRITER*' does either an action on channel *ComRW* or channel *FakeRW2*.

PROCESS and intruder modeling

PROCESS_New describes the processing procedure for the new access control, which is used to decrypt messages. It decrypts messages to check if the data is acquired correctly.

$$\begin{aligned}
&\mathrm{PROCESS_New()} =_{df}\\
&\quad (\mathrm{GetData?msg_{pro}.E(k', k1^{-1}).E(dk, data).E(k2, dk').k^{-1}}\\
&\quad \rightarrow \left(\begin{array}{l}(\mathrm{GetData!msg_{ack}.YES} \rightarrow \mathrm{SKIP})\\ \triangleleft((\mathrm{k'} == \mathrm{k})\&\&(\mathrm{k1} == \mathrm{k2})\&\&(\mathrm{dk} == \mathrm{dk'}))\triangleright\\ (\mathrm{GetData!msg_{ack}.NO} \rightarrow \mathrm{SKIP})\end{array}\right) \Box \mathrm{CHECKP()}; \mathrm{PROCESS()}
\end{aligned}$$

The set of facts that intruders might learn in the new access control is defined as *Fact'*.

$$\begin{aligned}
\mathrm{Fact'} =_{df}\ & \{\mathrm{R,W}\} \cup \{\mathrm{DKN,DN}\} \cup \mathrm{MSG'_{out}} \cup \{\mathrm{K}, \mathrm{K^{-1}}, \mathrm{DK}\}\\
& \cup \{\mathrm{E(key,content)} \mid \mathrm{key} \in \{\mathrm{K,DK}\}, \mathrm{content} \in \{\mathrm{Data,DK}, \mathrm{K^{-1}}\}\}
\end{aligned}$$

All the messages transmitted between entities can be intercepted by the intruder. It supports facts deduced from its known facts and messages faking to disturb normal communication between entities. The formalization of is *INTRUDER_New* as below.

$$\begin{aligned}
&\mathrm{INTUDER'(F)} =_{df}\\
&\quad \Box\Box_{\mathrm{m} \in \mathrm{MSG'_{out}}} \mathrm{FakeRW1?m} \rightarrow \mathrm{FakeRW2!m} \rightarrow \mathrm{INTRUDER'(F \cup Info(m))}\\
&\quad \Box\Box_{\mathrm{m} \in \mathrm{MSG'_{dat2}}} \mathrm{FakeRW2?m} \rightarrow \mathrm{FakeRW1!m[[k^{-1} \leftarrow K_{\mathit{I}-1}]]} \rightarrow \mathrm{INTRUDER'(F \cup Info(m))}\\
&\quad \Box\Box_{\mathrm{m} \in (\mathrm{MSG'_{out}} \setminus \mathrm{MSG'_{dat2}})} \mathrm{FakeRW2?m} \rightarrow \mathrm{FakeRW1!m} \rightarrow \mathrm{INTRUDER'(F \cup Info(m))}\\
&\quad \Box\Box_{\mathrm{f} \in \mathrm{Fact'}, \mathrm{f} \notin \mathrm{F}, \mathrm{F} \mapsto \mathrm{f}} \mathrm{Initialization\{l = false\}} \rightarrow \mathrm{Deduce.f.F} \rightarrow\\
&\qquad \left(\begin{array}{l}(\mathrm{DataLeakageSuccess\{l = true\}} \rightarrow \mathrm{INTRUDER'(F \cup \{f\})})\\ \triangleleft(\mathrm{f} == \mathrm{Data}) \triangleright (\mathrm{DataLeakageError\{l = false\}} \rightarrow\\ \mathrm{INTRUDER'(F \cup \{f\})})\end{array}\right)\\
&\mathrm{INTUDER_New} =_{df} \mathrm{INTRUDER'(IK')} \qquad \mathrm{IK'} =_{df} \{\mathrm{R,W,K,K}_{\mathit{I}}, \mathrm{K}_{\mathit{I}-1}\}
\end{aligned}$$

Please refer to the website(The full implementation of the innovative methods in PAT can be found in https://github.com/asunafy/SVF-NDN.) for full implementation of the innovative method.

Related work

There are several researches on formal methods for cyberattacks. Bernardeschi et al.[23] introduced a framework integrating formal verification and network simulation to evaluate attacks on wireless sensor networks. This approach is effective in simulating attack scenarios and verifying system robustness but is primarily designed for specific industrial domains, limiting its generalizability to other network architectures. Hóu et al.[24] combined digital twins and runtime verification to protect satellites systems from cyberattacks. However, its reliance on runtime data makes it less suitable for preemptive and formal security verification in broader contexts. Poorhadi et al.[25] presented a formal approach to evaluate the cyberattacks on the European rail traffic management system with Event-B. Yet, it lacks scalability for large-scale distributed networks. Sakata et al.[26] formalized the fallback control of the industrial control systems in UPPAAL and analyzed whether the supervisor can give incident response during cyberattacks. While it is adept at ensuring incident response mechanisms, it focuses on predefined scenarios and does not address adaptive attack patterns. Most of the above approaches excel

in specific industrial applications but fall short in addressing cyberattacks in more dynamic and distributed network environments such as access control in NDN.

Formal methods have been widely applied in the aspect of access controls. Wu et al.[27] formalized the Role-Compatibility Model and proved that if a resource is protected in a policy, then no existing sequence of events would compromise security. This method is robust in ensuring policy compliance but is limited by its static nature, making it less effective in adapting to dynamic policy changes. Hu et al.[28] defined a standardized structure for mandatory access control (MAC) mechanisms to provide property verification and automated generation of test cases. The approach efficiently automates test case generation, but its reliance on a specific modeling language reduces its flexibility for other access control paradigms. Ferrara et al.[29] combined abstraction and reduction to perform security analysis on administrative role-based access control models (RBAC). By abstracting policies into imperative programs, they effectively verify policy security, but this approach struggles with scalability in large systems with complex roles and hierarchies. These researches provide a solid foundation for formal access control verification but primarily focus on traditional access control models. In contrast, this paper specifically addresses the security concerns in NDN access control through data encryption and formal methods to handle cyberattacks.

There were also some studies focusing on the security of NDN and NDN-like environments, such as Information Centric Networking (ICN) and Content-centric networking (CCN). Fan et al.[30]presented a complete secure file transfer protocol for NDN called FTP-NDN, and gave formal security models and proofs for it. While comprehensive, it is designed for file transfer scenarios and does not address general access control issues. In[31], Viera et al. proposed a security protocol based on identity-based encryption to support ICN in smart grids, which enhances identity authentication but faces challenges in scalability for larger ICN environments. Wang et al.[32] proposed a session-based access control mechanism for ICN. Wood et al.[33] provided strong end-to-end content security for CNN using a combination of proxy re-encryption and identity-based encryption, but the dependency on proxy components introduces potential vulnerabilities. Most of these works propose effective mechanisms for specific use cases but lack a comprehensive framework for robust security verification. This paper addresses this gap by introducing a security verification framework for NDN access control that supports the enhancement of access control measures against evolving cyberattacks.

Conclusion

In this paper, a security verification framework for NDN access control (SVF-NDN) has been presented. It takes the access control scheme and security properties as inputs and conducts formal modeling and verification for evaluating the scheme's performance under cyberattacks. Taking data encryption-based NDN access control as a case study, several properties including deadlock freedom, data availability, key authentication, data leakage prevention, and data access protection have been verified. The verification results indicate that key authentication and data leakage protection are rendered invalid in the presence of intruders. As a result, two improvement methods and one innovative method were proposed. The improvement methods improved the access controls by adding digital signatures and digital certificates. The updated access control protected against data leakage prevention when the ACM is secure. The innovative method created a new access control instead to deal with the situation when ACM is invaded. The verification results showed the successful protection of data leakage prevention. Additionally, four algorithms were proposed to support the update of the CSP model in SFV-NDN. This work is significant for improving the security and robustness of NDN access control.

Future work

As for future work, the consideration of modeling and verifying alternative access control solutions in NDN using the framework SFV-NDN, as well as addressing other potential security issues related to NDN access control, will be the focus. Also, further research will be conducted on cyber attacks, including chain attacks and probabilistic attacks, to enhance the model's ability to respond to such attacks. Due to the significant importance of caching in the NDN network, the research will also focus on attacks targeting caches, equipping the model with the capability to effectively handle such attacks.

Data availability statement

All data generated or analysed during this study are included in this published article.

Received: 25 July 2024; Accepted: 31 January 2025
Published online: 14 February 2025

References

1. Zhang, L. *et al.* Named data networking (NDN) project. Tech. Rep. NDN-0001, PARC (2010).
2. Bari, M. F., Chowdhury, S. R., Ahmed, R., Boutaba, R. & Mathieu, B. A survey of naming and routing in information-centric networks. *IEEE Communications Magazine* **50**, 44–53 (2012).
3. Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D. & Ohlman, B. A survey of information-centric networking. *IEEE Communications Magazine* **50**, 26–36 (2012).
4. di Vimercati, S. D. C., Paraboschi, S. & Samarati, P. Access control: principles and solutions. *Softw. Pract. Exp.* **33**, 397–421 (2003).
5. Chen, T., Lei, K. & Xu, K. An encryption and probability based access control model for named data networking. In *IEEE 33rd International Performance Computing and Communications Conference, Austin, TX, USA, December 5-7*, 1–8 (2014).
6. Misra, S., Tourani, R. & Majd, N. E. Secure content delivery in information-centric networks: design, implementation, and analyses. In *Proceedings of the 3rd, 2013 ACM SIGCOMM Workshop on Information-Centric Networking, August 12, Hong Kong, China*, 73–78 (2013).
7. Golle, J. & Smetters, D. *Ccnx access control specifications* (Tech. Rep, Xerox Palo Alto Research Center-PARC, 2010).

8. Hamdane, B., Boussada, R., Elhdhili, M. E. & Fatmi, S. G. E. Towards a secure access to content in named data networking. In *26th IEEE International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises, Poznan, Poland, June 21-23*, 250–255 (2017).
9. Vistbakka, I. & Troubitsyna, E. Modelling and verification of safety of access control in SCADA systems. In *Risks and Security of Internet and Systems - 15th International Conference, Paris, France, November 4-6*, vol. 12528 of *Lecture Notes in Computer Science*, 354–364 (2020).
10. Rivera, V. Formal verification of access control model for my health record system. In *25th International Conference on Engineering of Complex Computer Systems, Singapore, October 28-31*, 21–30 (IEEE, 2020).
11. Vistbakka, I., Barash, M. & Troubitsyna, E. Towards creating a DSL facilitating modelling of dynamic access control in Event-B. In *Abstract State Machines, Alloy, B, TLA, VDM, and Z - 6th International Conference, Southampton, UK, June 5-8*, vol. 10817 of *Lecture Notes in Computer Science*, 386–391 (Springer, 2018).
12. Fei, Y. & Zhu, H. Modeling and verifying NDN access control using CSP. In *Formal Methods and Software Engineering - 20th International Conference on Formal Engineering Methods, Gold Coast, QLD, Australia, November 12-16*, 143–159 (2018).
13. Brookes, S. D., Hoare, C. A. R. & Roscoe, A. W. A theory of communicating sequential processes. *J. ACM* **31**, 560–599 (1984).
14. Hoare, C. A. R. *Communicating Sequential Processes* (Prentice-Hall, 1985).
15. Li, R., Yin, J., Zhu, H. & Vinh, P. C. Verification of rabbitmq with kerberos using timed automata. *Mob. Networks Appl.* **27**, 2049–2067 (2022).
16. Fei, Y., Zhu, H. & Yin, J. FVF-AKA: A formal verification framework of AKA protocols for multi-server iot. *Formal Aspects Comput.* **35**, 21:1–21:36 (2023).
17. Xu, J., Yin, J., Zhu, H. & Xiao, L. Formalization and verification of kafka messaging mechanism using CSP. *Comput. Sci. Inf. Syst.* **20**, 277–306 (2023).
18. Yin, J., Zhu, H. & Vinh, P. C. Formalization and analysis of haystack architecture from process algebra perspective. *Mob. Networks Appl.* **25**, 1125–1139 (2020).
19. PAT. PAT: Process analysis toolkit (2024).
20. Si, Y. et al. Model checking with fairness assumptions using PAT. *Frontiers Comput. Sci.* **8**, 1–16 (2014).
21. Sun, J. et al. Modeling and verifying hierarchical real-time systems using stateful timed CSP. *ACM Trans. Softw. Eng. Methodol.* **22**, 3 (2013).
22. Liu, Y., Sun, J. & Dong, J. S. Developing model checkers using PAT. In *ATVA*, 371–377 (2010).
23. Bernardeschi, C., Dini, G., Palmieri, M. & Racciatti, F. A framework for formal analysis and simulative evaluation of security attacks in wireless sensor networks. *J. Comput. Virol. Hacking Tech.* **17**, 249–263 (2021).
24. Hóu, Z., Li, Q., Foo, E., Dong, J. S. & de Souza, P. A digital twin runtime verification framework for protecting satellites systems from cyber attacks. In *26th International Conference on Engineering of Complex Computer Systems, Hiroshima, Japan, March 26-30*, 117–122 (IEEE, 2022).
25. Poorhadi, E., Troubitysna, E. & Dán, G. Formal modelling of the impact of cyber attacks on railway safety. In *Computer Safety, Reliability, and Security. SAFECOMP 2021 Workshops - DECSoS, MAPSOD, DepDevOps, USDAI, and WAISE, York, UK, September 7, 2021, Proceedings*, vol. 12853 of *Lecture Notes in Computer Science*, 117–127 (Springer, 2021).
26. Sakata, K., Fujita, S. & Sawada, K. Model verification of resilient third-party monitoring system against cyberattacks. In *IEEE International Conference on Consumer Electronics, Las Vegas, NV, USA, January 7-9*, 1–6 (IEEE, 2022).
27. Wu, C., Zhang, X. & Urban, C. A formal model and correctness proof for an access control policy framework. In *Certified Programs and Proofs - Third International Conference, CPP 2013, Melbourne, VIC, Australia, December 11-13, 2013, Proceedings*, 292–307 (2013).
28. Hu, V. C., Kuhn, D. R., Xie, T. & Hwang, J. Model checking for verification of mandatory access control models and properties. *International Journal of Software Engineering and Knowledge Engineering* **21**, 103–127 (2011).
29. Ferrara, A. L., Madhusudan, P. & Parlato, G. Security analysis of role-based access control through program verification. In *25th IEEE Computer Security Foundations Symposium, Cambridge, MA, USA, June 25-27*, 113–125 (2012).
30. Fan, C., Chen, I., Cheng, C., Huang, J. & Chen, W. FTP-NDN: file transfer protocol based on re-encryption for named data network supporting nondesignated receivers. *IEEE Systems Journal* **12**, 473–484 (2018).
31. Vieira, B. & Poll, E. A security protocol for information-centric networking in smart grids. In *Proceedings of the 2013 ACM Workshop on Smart Energy Grid Security, November 8, Berlin, Germany*, 1–10 (2013).
32. Wang, Y., Xu, M., Feng, Z., Li, Q. & Li, Q. Session-based access control in information-centric networks: Design and analyses. In *IEEE 33rd International Performance Computing and Communications Conference, Austin, TX, USA, December 5-7*, 1–8 (2014).
33. Wood, C. A. & Uzun, E. Flexible end-to-end content security in CCN. In *IEEE 11th Consumer Communications and Networking Conference*, 858–865 (2014).

Acknowledgements

This work was partially supported by the National Natural Science Foundation of China (No. 62372178), the Shanghai Key Laboratory of Trustworthy Computing (No. OP202003), the 2023 Shanghai Educational Science Research Program (No. C2023039), Specialized Program for Creating Accessible Spaces for Visually Impaired Students in Special Education (No. 115-AC9103-25-038003), New Young Teachers Program of Shanghai Jiao Tong University, and Northwestern Polytechnical University Taicang Yangtze River Delta Research Institute Innovation Workstation Open Fund.

Author contributions

Yuan Fei conceived the experiments, Jiaqi Yin and Lijun Yan conducted the experiments, Yuan Fei analysed the results. All authors reviewed the manuscript.

Declarations

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to L.Y.

Reprints and permissions information is available at www.nature.com/reprints.