Content Security in NDN using NAC

Larry Huang
Fulton Schools of Engineering
Arizona State University
Tempe, AZ, USA
lhuan130@asu.edu

***Abstract*—Named Data Networking's content security is a persistent problem with difficult layers. While the architecture's peer-to-peer design and in-network caching have many potential benefits over the existing internet's end-to-end paradigm, it comes with various security and control difficulties inherent to the distributed caching of content. This project tries to evaluate a specific method for content security involving distributed data owners and sources, Name-based Access Control, and demonstrates it using simplified scripts on FABRIC nodes.**

Keywords— NDN, content delivery networks, cryptography, FABRIC, content security, DRM

I. Introduction and Background

Named-Data Networking (NDN) was one of several projects the NSF funded in order to evaluate alternatives in the future of internet architecture and networking designs [8]. The basic principle draws on the dominant (by traffic volume) use case of the internet as used for content delivery (described in the predecessor project of Content-Centric Networking (CCN) first seen in 2006) to describe a networking paradigm centered around identifiable data chunks, in contrast to the existing internet's host-based network stack.

NDN envisions a network core with built-in content caching and forwarding based upon Names for content and Interest Requests without the end-to-end traversals of the existing internet. The great disadvantage of removing end-to-end from a networking paradigm is that additional measures have to be taken to secure the network against untrusted nodes, both on endpoints and within the distributed network core. To date, any attempt at a more secure implementation of NDN has been unable to fully distribute authority, requiring at least one exchange with specific nodes capable of authenticating requestors or data sources [7].

Furthermore, NDN's design came with three core deficiencies which have served as major roadblocks to more widespread adoption when first proposed for research:

1. Namespace management (the problem of which names are valid for what types and data, as well as who is trusted to name data) within NDN has a similar motivation to problems in the early internet that birthed the Domain Name System (DNS) hierarchies and services. The distributed design of NDN compounds the difficulty of securing namespaces by disincentivizing centralized trust and changing the paradigm of various security and availability problems.
2. Building/testing actual implementations of NDN requires significant investment in hardware resources for network nodes to support distributed in-network content caching.
3. The problem of content security becomes apparent when checking whether NDN's theoretical utility has practical value. How can a data owner or producer secure content once it is cached in the network?

In this paper, we focus on the third area. Efforts at designing mechanisms for content security in NDN all lead back to encryption to prevent unauthorized access, as is already used in the existing internet paradigm. The basic exchange of content security follows the same pattern as a secured HTTPS exchange: asymmetric key cryptography is used to securely exchange a symmetric key to avoid snooping of the key, before symmetrically encrypted data is used to communicate without being overhead.

In the existing internet paradigm, Digital Rights Management (DRM) uses similar methods to secure content. These small modules for web browsers (as defined in Encrypted Media Extensions (EME) [6]) or in content delivery client applications abstract the key exchanges away from user control, both to protect the keys from malicious users and to improve ease of use for non-technical users. While this idea is effective in our existing end-to-end paradigm, the per-hop nature of the NDN architecture would notably increase the delay and overhead of performing these exchanges given the cache-based interactions of a NDN network. As such, efforts in NDN content security were made to try and reduce the repetitiveness of encryption activities while still allowing content to be secured, leading to Name-based Access Control (NAC)

To show how NAC can implement the benefits of both existing DRM content security and NDN's distributed producer model, I used Python to implement a simple example of NAC in action. The JuPyter notebook provided allows you to run it in FABRIC (Summer 2024 version) to demonstrate the viability of separating access control keys from content keys in a content delivery scenario. Time constraints and certain quirks of the allowed network activity on FABRIC informed certain design decisions within the demo. Project is linked both as the first item in the reference list and as follows:

https://github.com/lhuan130/NACdemo_lhuan130

II. Related Work

A. Securing Network Caches

NDN was architected in the middle years of growth towards the modern landscape of content delivery networks and did not contain a complete design for content security. In the years since, there have been multiple efforts to design security features in NDN for content producers and consumers [7].

The basic idea behind access control within network caches follows the following core principles to establish content security (Figure 1 showing the exchanges between Producer and Consumer):

1. Only encrypted content is sent to network caches. Unencrypted content is only available on the producer node itself, or after decryption upon reaching a valid user.
2. A consumer who wants to access content must authenticate with the owner in order to receive the necessary keys.
3. All exchanges of keys should use asymmetric cryptography to prevent snooping. Symmetric cryptography is used on content for efficiency.

However, this design seems to lead to two specific pathways that each have problems. The first problem comes from using a single content key for all content by that producer. If the same content key is used for all content, then compromise of the content key compromises all the data from that producer currently in the network cache.

Using content keys that are specific to the data comes with a problem in the other direction. If an individual authorized to access content wants to access multiple items, then each user must perform a key exchange for each piece of content by that producer, which becomes inefficient at scale in content delivery. This is further complicated by a reduced ability to target individual nodes in NDN: performing these exchanges would supposedly happen through the same interest and cache forwarding mechanisms as the content itself in NDN.

The best design would likely allow some method of grouping content keys based upon data that is meant to be accessed together. This would enable producers to only require a single key exchange to provide access to a specific scope or content group, balancing granularity of control and security.

B. NDN in FABRIC

The FABRIC testbed provides the networking hardware with which we will demonstrate test the viability of content revoking on network caches. FABRIC is useful for NDN testing for several reasons: for one, the software-defined networking capabilities and

There have been previous efforts to demonstrate the viability of NDN on the testbed [2, 5], with various successor projects on the testbed evaluating various aspects of NDN or attempting to address shortcomings in the architecture's security or design. However, the implementation of NDN itself in FABRIC is not the core focus of this project.

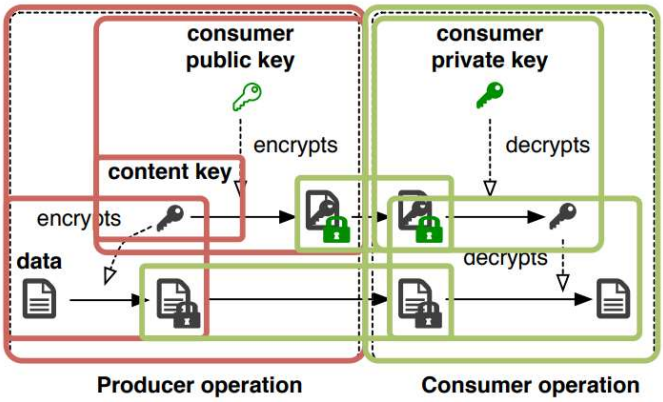

Fig. 1. From [4]. The basic idea within content security. The content key is only distributed to allowed clients. Asymmetric cryptography is used to secure the key exchange between Producer and Consumer.

Gaouette and McMillion [5] created a working example of the content key exchange shown in Figure 1 within a simple NDN topology (Figure 2) on FABRIC, demonstrating the viability of a simple content key exchange (the first half of NAC, using content keys). Their topology uses two NDN nodes to perform the exchange through a direct link.

While their demo provided a helpful starting point, there are a few problems to consider when considering this as an example of NAC in NDN. First, NDN envisions very few scenarios in which you need to request content directly from the producer server; network caches should be the source of encrypted content delivery unless it has not yet been cached closer to a requesting user due to lack of interest. Second, the single hop between a key provider and consumer provided by their experiment would be less likely in a real-world NDN scenario. For this reason, the topologies within this experiment use slightly longer routing paths to simulate an actual cache on the NDN path.

Furthermore, something that has changed in FABRIC between their project's development version (code committed in May 2023; version released before then) and the current stable version (Summer 2024) causes it to no longer be functional without modification. These changes have also made starting NDN-DPDK impossible as it is; therefore, their report is the only example left.

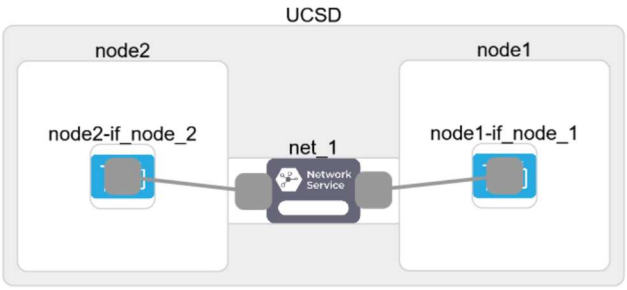

Fig. 2. FABRIC NAC example topology from [5], with two NDN nodes performing direct queries and content or key exchanges.

C. Group (Namespace) Keys in NAC

Name-based Access Control is a collective of ideas that have attempted to build secure channels for content producers to provide access-controlled content across a NDN network. NAC designs are being studied by the current maintainers of NDN designs and literature, who are the active developers of named data networking; code examples are provided for examination and learning by new participants [2].

NAC solves the issue with balancing granularity and security for content keys by adding an asymmetrical layer to content key access [3, 4]. The first stage of this control requires that there be two control layers for producing keys.

1. Data Owner decides how to organize the namespace of data/content availability and produces an asymmetric key pair referred to as the Group encryption and decryption key.
2. Producer creates the symmetric content key(s) organized in a way defined by the data owner (as granular as is useful for access control).

This separation simplifies the content Producer's role significantly while expanding its capabilities. The distribution of data production is simplified because the encrypted content key can now be bundled with encrypted content without compromising access control; all a Producer node needs is the appropriate Group encryption (public) key. Content keys do not have to be protected by individual producers maintaining access control data; instead, the content key is protected by the Data Owner's defined. Access Control management is now the responsibility of the Data Owner, who authenticates access to the Group decryption (private) keys. All content keys that are valid within a Group should be encrypted with the same Group encryption key; if a consumer is allowed access to all that data, they can use the Group decryption key to access all unique content keys provided by the Producer under that namespace.

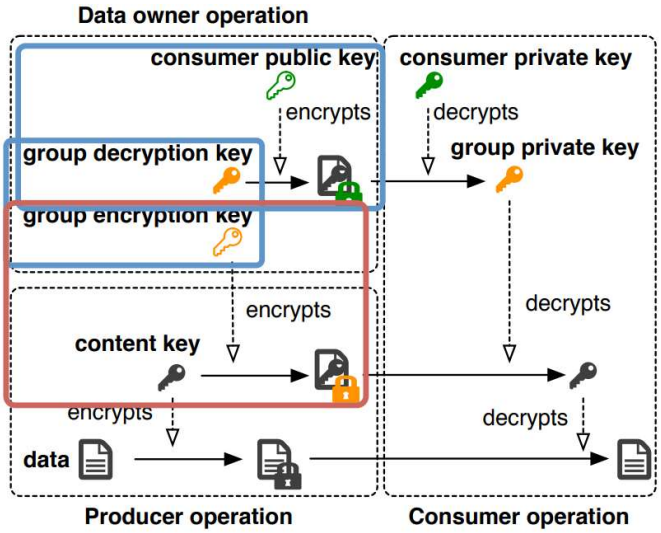

Fig. 3. NAC group key usage as displayed in [4]. Note that the content key itself does not need an additional layer of encryption in the Producer-Consumer exchange.

Like Figure 1, Figure 3 demonstrates the interactions between Data Owner, data Producer, and a Consumer client for NAC-protected content access. This is the interaction modeled in the attached software project [1].

III. Experimental Setup

A. Previous Efforts (NDN-DPDK demo)

In this experiment's original design, a content producer would have deployed content to a NDN network node for distribution. Figure 4 provides a practical example of a topology with two users nodes.

The original plan was to produce a working implementation in NDN-DPDK on FABRIC based upon previous student projects [2, 5]. However, several problems existed with this plan:

1. NDN-DPDK is resource-intensive, requires specific hardware interfaces, and configuration required way more time than expected for correct function.
2. Previous student examples no longer worked on the current FABRIC server/library versions, only providing information about required supporting software packages and resource requirements (insufficient for successful build, installation, and configuration).

Therefore, the efforts described in the following section have replaced an NDN-DPDK implementation for demonstrating NAC on FABRIC.

B. Current Efforts (Group key exchange demo)

To simplify this project, we will skip the process of namespace definition so that requests can be more easily coded into a successful demonstration. In addition, I have refrained from implementing a naming hierarchy and delivering well-designed name packets under key-exchange conventions as described in the NAC tutorials and papers [3, 4].

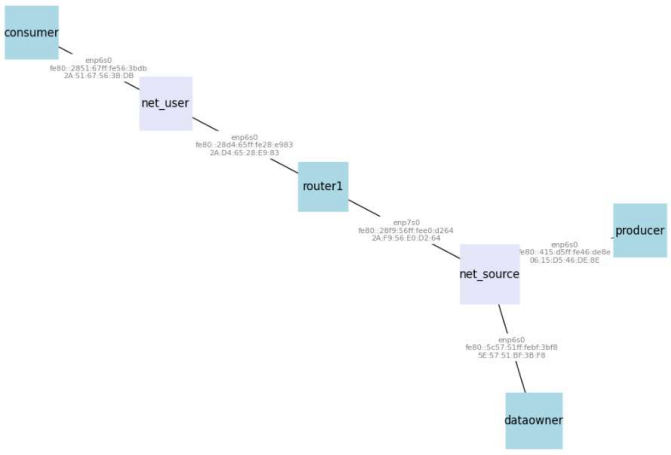

Fig. 4. The demonstration's topology with the NDN router node surrounded by NDN users displaying the Data Owner, Producer, and Consumer.

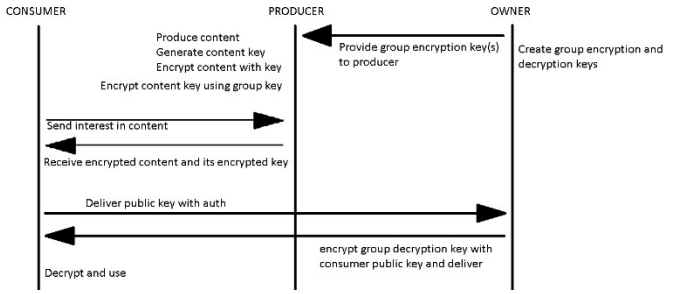

Fig. 5. The experiment's simplified NAC Group key exchange interactions, in a diagram by the order of operations. Not included are any intermediate NDN routing nodes, as there is only a simulated router node in the demonstration itself.

For this demonstration project demo, the JuPyter notebook provides a simple demonstration of how NDN nodes and users would perform a NAC group key operation using four FABRIC nodes. In the topology (shown in Figure 4), we have the nodes existing in two subnets; one contains the data owner and producer, while the other contains the consumer. To simplify deployment, all nodes deployed are kept on the same FABRIC site. During development, it was discovered that sending files back and forth among nodes would require a lot more configuration for HTTP servers using Apache2 than desired; as such, the JuPyter notebook provides an step-by-step procedure for the JuPyter server to replace the "router node" in the demo; the results are cached in a specified repo folder and sent using FABlib commands instead of between nodes directly using Python. The procedure and/or role of each node is described as follows:

1. DATA OWNER (procedure)
   a. Generate new group encryption key
   b. Wait for request for group encryption key (from Producer) and reply with it
   c. Wait for request for group decryption key (from Consumer) and use the included public key to encrypt; reply with encrypted group decryption key
2. PRODUCER (procedure, started after Data Owner is ready)
   a. Generate new AES content key
   b. Encrypt content with content key, send encrypted content to router
   c. Send request to Data Owner for group encryption key
   d. Encrypt content key with group encryption key, send encrypted content key to router
3. ROUTER responds to the following requests:
   a. Forward
      i. Request for group encryption key (Producer to Owner)
      ii. Decryption key request (Consumer to Owner)
      iii. Decryption key reply (Owner to Consumer)
   b. Store content pushed from Producer (encrypted content and content key)
4. CONSUMER (procedure)
   a. Send interest for content to router (receive encrypted content and encrypted content key)
   b. Send authentication request containing own public key to data owner
   c. Receive group decryption key
   d. Decrypt group decryption key with own private key
   e. Decrypt content key using group decryption key
   f. Decrypt content using content key

An unencrypted content file will be stored by the content producer node but will not be provided to the router node's cache. Instead, the content producer node will provide the NDN network an encrypted copy of the content and an encrypted copy of the key used to encrypt it.

The scripts provided are started in order: the Data Owner generates the Group key pair, the Producer takes the Group encryption key and stores the encrypted content and encrypted content key on the Router, then the Consumer begins its request procedure to eventually acquire the unencrypted content.

IV. Results

Figure 5 shows the order of actions in the interaction in a more abstracted view. Organizing the demonstration this way enables us to examine the actual interactions involved in NAC, without the complexity of an NDN system interfering (or, in the other direction, demonstrating the security design's utility).

Difficulties encountered in implementing the group key interactions in actual NDN nodes given issues with NDN libraries such as NDN-DPDK prevent practical metrics from being gathered. While I could have set up cursory timing for the key interactions, any actual metrics would fall under three categories: time to access content from interest, viability of access, and the time required to provide access to past or future content by pre-creating or storing keys. None of these features are available in the extremely simplified demo that was actually produced. As such, there are no tangible metrics to compare.

Still, while this demo only uses a single group key to demonstrate viability, we can extrapolate some further understanding based upon what is encrypted-and-decrypted. Using the content key as the first barrier to entry provides the first principle of access control, deny-as-default, to content security. The content key is then secured using the Group Encryption Key. Layering the content key and Group key provides both granularity and

V. Summary, Conclusions, and Future Work

The original goal of this project was to build on FABRIC NAC's NDN wrapper to demonstrate the viability of NAC group namespaces in NDN, with the intention of measuring the performance hit that added security exchanges and rounds of encryption/decryption would cause in NDN. However, this goal fell through due to the difficulty of configuring NDN-DPDK to work in FABRIC's current version. As such, the design pivoted to a simpler implementation of the group key exchange to display its base functionality in enhancing content security under an additional layer of asymmetric encryption.

Still, this super-simplified demonstration does meet all the requirements of a Name-based Group Key Exchange as defined in [3] and [4]. In a team with more experience or understanding

when it comes to managing NDN and/or DPDK's configuration, perhaps more progress could have been made towards an actual NDN implementation. The demonstration is included within a JuPyter notebook; scripts utilized by member nodes created on FABRIC using that notebook are also included in the repository [1].

Future work in this area would actually take the time to try and fix the existing FABRIC deployment of NDN-DPDK in order to actually perform these exchanges using NDN nodes. This would also allow us to take actual metrics of an NDN implementation's content delivery performance using FABRIC's hardware resources. In such a design, we could simultaneously evaluate the practicality of key exchanges over longer distances or when delivering security exchanges through multiple NDN nodes between the consumer and data owner endpoints.

REFERENCES

[1] L. Huang. "Name-based Access Control Group Key Demonstration" [https://github.com/lhuan130/NACdemo_lhuan130]

[2] A. Nair, T. Sinkinson, J. Womack, Y. Yuan. "Implementing Named Data Networking on FABRIC" [https://github.com/initialguess/fabric-ndn/blob/main/NDN%20on%20FABRIC.pdf]

[3] A. Afanaysev et. al. "NAC: Name-Based Access Control Library for NDN" [https://github.com/named-data/name-based-access-control]

[4] Y. Yu. "Name-Based Access Control NDN Tutorial." In Proceedings of ACM Conference on Information-Centric Networking, Sep. 2015. [https://named-data.net/wp-content/uploads/2015/11/ndn-tutorial-named-based-access.pdf]

[5] A Gaouette, B McMillin. "Exploring Named Data Networking and its Security Considerations on FABRIC." [https://github.com/agaouett/fabric_nac/blob/main/fabric-ndn-security.pdf]

[6] World Wide Web Consortium. "Encrypted Media Extensions." W3C Working Draft, Aug. 2024. [https://www.w3.org/TR/encrypted-media-2]

[7] D. Kim, J. Bi, A. V. Vasilakos and I. Yeom, "Security of Cached Content in NDN," in IEEE Transactions on Information Forensics and Security, vol. 12, no. 12, pp. 2933-2944, Dec. 2017, doi: 10.1109/TIFS.2017.2725229.

[8] L. Zhang et. al. "Named Data Networking," ACM SIGCOMM Computer Communication Review, Vol. 44, Issue 3, pp. 66-73, Jul. 2014, doi: 10.1145/ /2656877.2656887.