An Overview of Security Support in Named Data Networking

Zhiyi Zhang, Yingdi Yu, Haitao Zhang, Eric Newberry, Spyridon Mastorakis, Yanbiao Li, Alexander Afanasyev, Lixia Zhang

***Abstract*—This paper presents an overview of the security mechanisms in the Named Data Networking (NDN) architecture that have been developed over the past seven years. NDN changes the communication model from the delivery of packets between hosts identified by IP addresses, as seen in IP, to the retrieval of named and secured data packets. Consequently, NDN fundamentally changes the approach to network security. Making named data the centerpiece of the architecture leads to a new security framework that: (i) secures data directly and (ii) uses name semantics to allow applications to reason about security. In this paper, we introduce NDN's approach to security bootstrapping, data authenticity, integrity, confidentiality, and availability.**

***Index Terms*—Named Data Networking, Security**

I. Introduction

Named Data Networking (NDN), a proposed Internet architecture, changes the basic network communication model. Instead of delivering IP packets to receivers identified by IP addresses, NDN lets consumers request the desired data by name. Naming data enables NDN to secure *data* directly at the network layer by making every data packet verifiable and, optionally, confidential.

In this paper, we provide an overview of the security framework in NDN, introducing the mechanisms with example prototype realizations and showing how all the components of the framework function together. We assume that the reader has some basic knowledge of cryptographic security, but is not necessarily familiar with the NDN architecture.

The paper is organized as follows: Section II provides a background on the NDN architecture and introduces the concepts upon which the security mechanisms are built. Section III describes the building blocks of NDN security and introduces an example application that will be used throughout the paper to help illustrate the presented security mechanisms. Section IV introduces the NDN security bootstrapping process. In Sections V, VI, and VII, we explain how security in NDN provides data authenticity, integrity, confidentiality, and availability. Throughout this paper, we aim to explain how NDN allows data to remain secure independent of a communication channel, as well as how it allows applications to validate received Data packets independent of how or from where the data was fetched. Moreover, we illustrate how applications can utilize name semantics to reason about the use of cryptographic keys based upon trust relations, instead of blindly relying on third-party certificate services. Section VIII discusses the basic differences between network security solutions in TCP/IP and NDN, illustrating how different network architectures lead to different security solutions; it also identifies remaining NDN security challenges.

We hope that this paper can serve as a guide to NDN security efforts for readers interested in NDN research, as well as a useful demonstration of new approaches to network security that differ from today's common practices.

Zhiyi Zhang, Yingdi Yu, Haitao Zhang, Spyridon Mastorakis, Yanbiao Li, and Lixia Zhang are with the Department of Computer Science, UCLA - e-mail: zhiyi, yingdi, haitao, mastorakis, lybmath, lixia@cs.ucla.edu.

Eric Newberry is with the Department of Computer Science, The University of Arizona - e-mail: enewberry@cs.arizona.edu

Alexander Afanasyev is with the Department of Computer Science, Florida International University - e-mail: aa@cs.fiu.edu

II. Background

In this section, we give a brief description of the basic concepts in an NDN network, followed by a high-level picture of NDN's security support framework, which will be further discussed in Section III.

A. Named Data Networking (NDN)

From 10,000 feet, one could view the basic idea of NDN as shifting HTTP's request (for a named data object)-and-response (with the object) semantics to the *network layer* [1]. Being a network-layer protocol, this request/response communication pattern works at a network packet granularity – each request, carried in an NDN *Interest* packet containing the name of the requested data, fetches one NDN *Data* packet (Figure 1); neither type of packet contains addresses. Applications running over NDN that produce data are known as *producers*, while those requesting data are known as *consumers*.

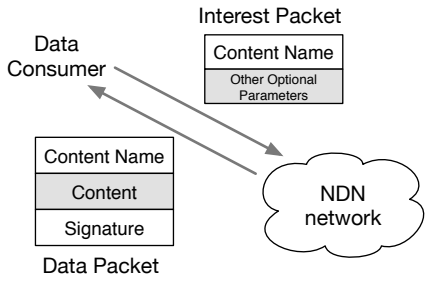

Fig. 1. Interest and Data packet in NDN

An NDN network runs routing protocol(s) to propagate the reachability of data name prefixes, similar to how IP networks use routing protocols to propagate IP address prefixes. Each NDN router forwards Interests based on their names, recording

the interfaces from which Interests are received and to which they are forwarded in a "Pending Interest Table" (PIT). Once an Interest packet reaches a matching Data packet, the latter will follow the reverse path of the corresponding Interest to reach the consumer, satisfying the corresponding PIT entry on each router along the path; the Data packet can also be catched in routers to serve future requests for the same Data packet. This stateful forwarding mechanism creates a closed feedback loop, enabling routers to make informed Interest forwarding decisions based on observed performance.

Following the earlier analogy, we do note that NDN Data packets, beside being network layer packets, also differ from HTTP data objects in two other important aspects: (i) all NDN Data packets are immutable; if producers want to change Data packet content, they must generate new packets with new names; and (ii) every NDN Data packet carries a security signature, generated by its producer's key at the time of data creation, cryptographically binding its name to its content; the packet may also be encrypted as needed. Named, secured data packets provide a basic building block for securing NDN communications. This also requires that all (data producing) applications possess cryptographic keys, which can be further validated when a consumer verifies a received Data packet, as explained below.

B. NDN Security: Basic Concepts

The NDN security framework is built on public-key cryptography. We call applications, and all other network communication participants in an NDN network, **entities**. Each entity possesses one or more names plus one or more cryptographic public-private key pairs.[1] An NDN certificate for a user named "`/ucla/cs/zhiyi`" binds this name and its key(s) together; it certifies the user's ownership of the name, along with their key. Each certified name is called an **identity**, and each entity can issue certificates for the sub-namespaces that it delegates to other entities.

Utilizing public-key cryptography to validate communications requires NDN to address the following three challenges:

Establishing trust anchor(s): In NDN, a trust anchor is the certificate authority for a given namespace. When a trust anchor is installed, users can verify other entities' signatures by backtracking and verifying the certificates along the certificate chain to the trust anchor. Trust anchors are usually installed on various entities via out-of-band mechanisms, and the development of these supporting mechanisms directly depends on the trust anchor model in use, as we discuss in the next section.

Providing effective solutions for trust management: To validate all received data, a participant must know which key(s) can legitimately sign or encrypt which pieces of data. Effective solutions must enable applications to express their own trust policies, and to execute these policies automatically.

[1]An entity can be a country, a university, a company, a neighborhood, a home, a user, a node, or an app process. The task of allocating names to entities is beyond the scope of the NDN design, just like the task of assigning IP addresses is beyond the scope of the TCP/IP design.

Providing usable key management solutions: Signing, verification, encryption, and decryption all involve cryptographic keys. Usable cryptographic solutions require mechanisms to assign and deliver proper keys or certificates in a secure, efficient, and *automatic* manner.

III. An Overview of Security in NDN

NDN utilizes a different trust anchor model than those commonly in use today, namely: (i) utilizing hundreds of commercial certificate authorities as trust anchors to authenticate other parties one wishes to communicate with (e.g., TLS certificates), (ii) installing a single global trust anchor (e.g., DNSSEC), and (iii) establishing trust in an ad-hoc way (e.g. trust on first use, or "TOFU"). NDN assumes that the authority of each networked system (an organization, a smart home, etc.) should establish its own local trust anchor(s). The entities under that authority can discover these trust anchors through local system settings, and then obtain certificates and learn trust policies from them. This trust model follows that of the Simple Distributed Security Infrastructure (SDSI) [2] in trust anchor establishment. With installed trust anchors, certificates, and trust policies, an entity can utilize its digital keys to ensure data authenticity, integrity, and confidentiality. In addition, NDN's in-network storage also helps improve data availability.

A. Building Blocks of Security in NDN

NDN security utilizes public-key cryptography and relies on the use of **digital keys**. In addition to keys, NDN also uses the following building blocks: trust policies and NDN certificates.

Trust Policies: Applications define trust policies to determine whether a given packet or identity is trustworthy or not. Given that data producers name Data packets (including certificates) in a structured and meaningful way, consumers can only accept packets with proper name formatting. Trust policies in NDN are based on name semantics; see more in Section V-A.

NDN Certificates: A certificate signer either signs NDN certificates under one's own namespace, or signs other keys (under different namespaces) as an endorsement (e.g. in a web of trust). An NDN certificate is a Data packet that carries public key information and can be fetched using normal Interest packets. Certificate names follow the naming convention "`/<prefix>/KEY/<key-id>/<issuer-info>/<cert-version>`", where the "prefix" represents an identity and the components after "KEY" are the key id, issuer information, and certificate version.

B. NDNFit as an Example

To aid the reader's comprehension, we use NDNFit [3], a prototype application for tracking and sharing personal fitness activity, as a specific example to illustrate NDN security mechanisms. NDNFit handles sensitive user information and thus requires strong data authenticity, integrity, and confidentiality.

As an example of a typical use case, a data owner, "Alice", wants to use NDNFit to learn her everyday fitness information. Alice has an app "Sensor" on her mobile phone and an app

"Analyzer" on her laptop. "Sensor" collects Alice's everyday time-location information while "Analyzer" generates inferred insights and visualized results by analyzing information provided by "Sensor". Figure 2 shows how this NDNFit system works.

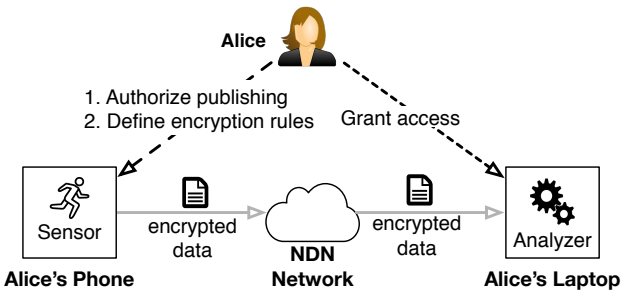

Fig. 2. NDNFit application workflow

To provide authenticity and integrity, NDNFit, as a health data application, requires that all data produced by "Sensor" and "Analyzer" be authenticable and provide for easy alteration detection, preventing unauthorized producers from producing fake data. To provide confidentiality, Alice only grants "Analyzer" the privilege to access the private fitness data produced by "Sensor" – no one else should be able to read this data. The rest of the paper will show how the security mechanisms in NDN help NDNFit achieve these objectives.

IV. Security Bootstrapping in NDN

The NDNFit system needs to be initialized by security bootstrapping before it can function. Security bootstrapping in NDN is the process by which entities obtain trust anchors, certificates, and learn trust policies. Here, Alice controls the whole system, and thus the trust anchor is Alice's certificate, which we assume to be "`/ndnfit/alice/KEY/key001/ndnfit-agent/version`".

A. Obtaining Trust Anchors

An entity needs trust anchors to identify authentic entities; at the very least, it should trust the certificate signer that issues certificates to these entities. Trust anchors are expected to either be pre-configured or securely obtained through some means, e.g., an out-of-band operation. Similar to SDSI, different system may have their own trust anchors and nodes within these systems can have their own means to obtain trust anchors. Thus, the process of obtaining trust anchors can be localized or distributed in NDN.

In our case, a naïve approach to trust anchor establishment would be to manually install Alice's certificate to "Sensor" and "Analyzer".

B. Obtaining Certificates

To generate Data packets with legitimate names and verifiable signatures, an application (producer) needs to obtain a name and a certificate for that name. For instance, in order to let other applications authenticate the NDN Data "`/ndnfit/alice/sensor/example`" produced by "Sensor", "Sensor" needs a certificate for the name "`/ndnfit/alice/sensor`". With trust anchors, an entity could apply for a certificate from a trusted certificate signer, which can be processed either manually or through automated means, e.g., NDN certificate management system (NDNCERT) [4]. NDN security grants flexibility to application developers to decide their own trust anchors. Depending on the system design, an application may obtain certificate from its own centralized certificate service, e.g., cloud-based applications, while a distributed application, e.g., p2p applications, may obtain certificates from its user or host node.

In NDNFit, the system trust anchor, Alice's certificate, resides in an NDNCERT daemon (called an agent) on her laptop, with this agent playing the role of the certificate signer. "Sensor" and "Analyzer" use the NDNCERT protocol to apply for certificates from this agent, and Alice will use the agent to verify the two apps using customized out-of-band challenges. Two certificates, "`/ndnfit/alice/sensor/KEY/1/alice-agent/version`" and "`/ndnfit/alice/analyzer/KEY/1/alice-agent/version`", will be issued to "Sensor" and "Analyzer", respectively. Note that it is Alice who controls the namespace "`/ndnfit/alice`" and can determine the app names.

C. Learning Trust Policies

To determine whether an incoming packet is trustworthy or not, an application (consumer) needs to learn trust policies through out-of-band operations or inline communications after obtaining the trust anchor. In the first case, application developers can pre-configure trust policies in applications and may allow users to update trust policies later via user interfaces. The second case is preferable to the first: after obtaining the trust anchor, the application can fetch, verify, and learn trust polices (carried in Data packet(s)) from trusted sources, e.g., a cloud-based application can learn policies from its central server. When validating Data packets that carry trust policies, an application either simply verifies the signature value or uses its default trust policy (e.g., the Data must be directly signed by a trust anchor), if any.

In NDNFit, applications have pre-configured trust polices and rely on Alice to update the configuration as needed.

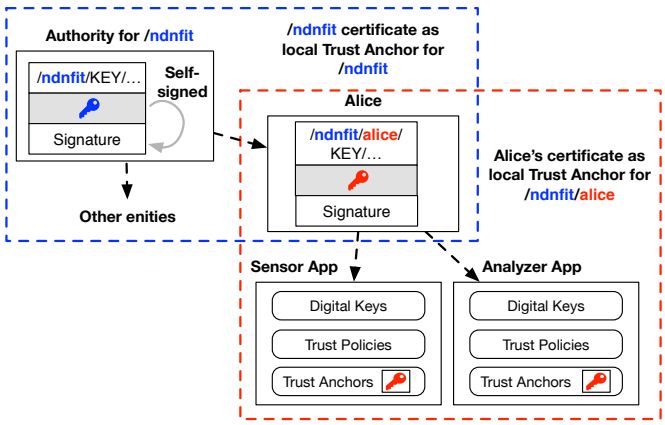

Fig. 3. Entities and Trust Anchors in NDN

After security bootstrapping, as shown in Figure 3, both "Sensor" and "Analyzer" will trust Alice and each will have their own trust policies and certificate under "`/ndnfit/alice`". The security bootstrapping of Alice's certificate takes place in a different network system where the trust anchor is "`/ndnfit/KEY/...`". Alice learns of this trust anchor and obtains the certificate "`/ndnfit/alice/KEY/...`" from the authority of the namespace "`/ndnfit`" (we will not go into the details of this process).

V. Authenticity and Integrity

Possessing certificates issued by the certificate signer, "Sensor" and "Analyzer" can produce Data packets within their corresponding namespaces and sign them using their corresponding private keys. In this section, we show how NDN security helps NDNFit ensure data authenticity and integrity.

NDN requires producers to sign every Data packet, enabling consumers to verify each incoming Data's signature, hence ensuring data authenticity and integrity. More importantly, NDN's rich name semantics enable consumers to use name-based trust policies to reason about trust by checking which piece of data is signed by which key. In this way, trust policies limit the power of each signing key and ensure that each trustworthy packet is signed by a legitimate key, providing data authenticity at a fine granularity. For instance, in NDNFit, the key certified in certificate "`/ndnfit/alice/sensor/KEY/1/alice-agent/version`" is only allowed to sign packets under prefix "`/ndnfit/alice/sensor`".

The authenticity and integrity of incoming Data packets (including certificates) is determined by a combination of two main factors:

Validation by Trust Polices: The structured naming conventions of Data packets and keys provide explicit and meaningful contexts for applications, enabling NDN applications to define rules that only accept packets with the desired name format and relationship between the name of a packet and the name of its signing key. To be more specific, the packet name, the signing key name, the relationship between these two names, and the trust anchor name must follow these rules. NDN's "Trust Schema" [5] is a realization of present name-based trust policies. See more in Section V-A.

Signature Verification: To verify a data signature, consumers retrieve the certificate of the corresponding producer, who is identified by the key name in a dedicated section of the Data packet. This certificate will recursively point to its signer's certificate and finally arrive at an anchor. The origin packet is considered to be valid if all fetched certificates, including the anchor, have valid signatures and can satisfy the trust policies.

A. Presenting Trust Policies using Trust Schemas

Specifically, trust schema make use of NDN's naming conventions to enable systematic descriptions of trust policies, namely: (i) how Data packet names are expected to be structured, (ii) how packet signing key names are expected to be structured, (iii) how Data packet names are expected to be related to signing key names, and (iv) which trust anchors are acceptable.

Upon receiving a packet, a consumer application uses trust schemas to assess the packet's trustworthiness before any cryptographic signature verification is performed. For instance, as shown in Figure 4, in addition to "Alice" ("`/ndnfit/alice`"), there is an entity "Bob" ("`/ndnfit/bob`"), who is also running an NDNFit system. We assume that both Alice's and Bob's certificates are signed by the same anchor certificate "`/ndnfit/KEY/key2007/self/version`". Both Alice's and Bob's devices produce data packets under their own prefixes, namely "`/ndnfit/alice/sensor/example`" and "`/ndnfit/bob/sensor/example`". As shown, there are two trust schemas. Schema "rule 1" accepts Data packets whose (i) name prefix is "`/ndnfit/alice`", (ii) signing key name prefix is "`/ndnfit/alice/KEY`", and (iii) certificate chain ends with the trust anchor "`/ndnfit/alice`". Accordingly, only packets signed by Alice and strictly under Alice's prefix are accepted. In contrast, "rule 2" has a looser requirement: all packets with the name and key name prefix "`/ndnfit`" and eventually signed by "`/ndnfit`" are considered trustworthy. As a consequence, "rule 2" accepts packets produced by both Alice's and Bob's devices.

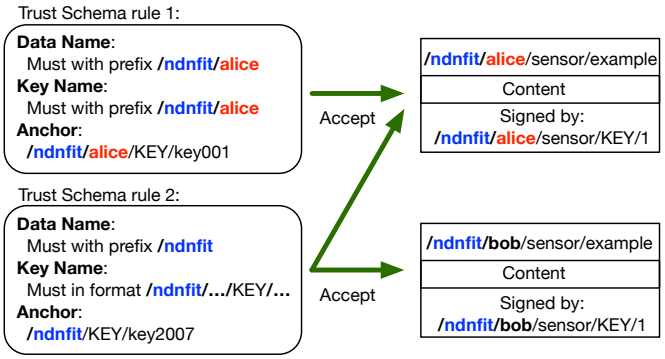

Fig. 4. An example of Trust Schema

B. Signed Interests

Interest packets can also be signed. For example, in an IoT scenario, when receiving an Interest packet containing a command, a smarthome device may need to authenticate the sender of the Interest before executing the command. To accomplish this, a controller can send a signed Interest to control IoT devices. The NDN Interest signature validation process is the same as the one used to validate Data packets.

VI. Data Confidentiality

NDNFit requires data confidentiality and access control to protect sensitive user information. When confidentiality is implemented via encryption, it is necessary to have a usable key distribution system to ensure that the involved entities can identify and fetch the relevant encryption and decryption keys.

For active point-to-point sessions, key exchange protocols like Diffie-Hellman [6] can derive encryption keys for the session, with both sides of the communication well-aware

of the key information. When sharing data with multiple parties (as in NDNFit), Diffie-Hellman may not be feasible or efficient. In these scenarios, the owner of the data needs to deliver decryption keys to authorized entities. In NDN, since Data names are structured and convey rich semantics, the NDN team is exploring approaches that leverage systematic naming conventions to inform consumers about how to fetch the corresponding decryption keys. These approaches would also automate the key distribution process and improve the protocols' usability.

For the second scenario, Named-based Access Control (NAC) [7] and its variants (such as NAC-ABE), schematized access control [8], and other protocols have been proposed. The NDNFit system uses NAC to achieve confidentiality and access control.

A. Name-based Access Control

In the NDNFit system, Alice is the data owner for all Data packets under "`/ndnfit/alice`" and determines who can access the confidential data and under what conditions. In NAC, each encryption key name will be explicitly appended to the name of the corresponding Data packet. For instance, a Data packet produced by "Sensor" has the name "`/ndnfit/alice/sensor/data/ENCRYPTED-BY /ndnfit/alice/E-KEY/sensor`", where the components after "`ENCRYPTED-BY`" are the encryption key name. As mentioned, "Analyzer" is authorized by Alice to access "Sensor"'s data under prefix "`/ndnfit/alice/sensor`". A simplified data production and encryption process is illustrated below.

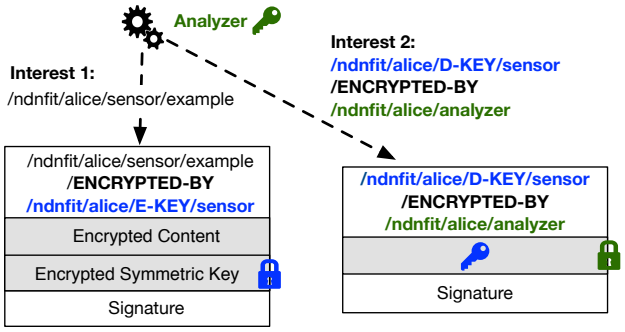

Fig. 5. Name-based Access Control in NDNFit

Key Generation: Alice will first generate a key pair ("E-KEY", "D-KEY") for encryption and decryption, respectively. She then produces two Data packets carrying "E-KEY" in plaintext and "D-KEY" encrypted by "Analyzer"'s public key. The "E-KEY" packet name follows the format "`/ndnfit/alice/E-KEY/sensor`", while the "D-KEY" packet name follows the format "`/ndnfit/alice/D-KEY/analyzer /ENCRYPTED-BY/ndnfit/alice/analyzer`".

Data Production: When producing data, "Sensor" first generates a symmetric key for content encryption. Then, it fetches "E-KEY" and encrypts the symmetric key with it. Finally, it packs the encrypted symmetric key into a Data whose name is "`/ndnfit/alice/sensor/data/ENCRYPTED-BY /ndnfit/alice/E-KEY/sensor`".

Data Consumption: As shown in Figure 5, when "Analyzer" wants to consume this Data, it first fetches the Data packet; the Data name conveys that the content is encrypted by the "E-KEY". To decrypt the content, "Analyzer" then fetches the corresponding "D-KEY". Notice that the fetched "D-KEY" is actually encrypted by "Analyzer"'s own key. By decrypting the content in the fetched "D-KEY" Data, the application obtains "D-KEY" and can decrypt the symmetric key and use the symmetric key to finally decrypt the content.

B. Fine Granularity through Key Naming

By defining naming conventions, NAC enables fine-grained access control. For instance, a "D-KEY" name could be "`/ndnfit/alice/D-KEY/sensor/example/monday`", indicating that the key will only be used to protect the data produced by "Sensor" on Mondays.

VII. Data and Certificate Availability

A. Data Availability by In-network Storage

The NDN architecture provides applications with high data availability because it is centered on content distribution. In NDN, named data is secured regardless of its location; that is, Data packets can be retrieved from in-network caches or any other storage system, no matter whether these cache services and storage systems are trustworthy or not. All capable forwarders can cache Data packets and popular Data packets are often cached in the network. Moreover, it is simple for producers to utilize long-term storage services (e.g., managed data repositories). In the NDNFit example (Figure 2), the in-network storage services in NDN network will help to improve the availability of Data produced by "Sensor".

B. Certificate Availability

The fact that certificates are a building block of NDN security makes certificate availability of vital importance. We claim that NDN certificates, as Data packets, also benefit from in-network caching and storage as discussed above. In order to further improve certificate availability, the NDN certificate bundle [9] has been designed. Certificate bundles enable producers to aggressively collect all the certificates in the certificate chain and pack them together as a bundle, allowing them make the whole chain of certificates available to consumers and avoiding any signature verification failures due to unreachable certificates. In NDNFit, "Sensor" produces Data packets and can prepare all needed certificates as a certificate bundle. Specifically, the bundle will contain the application certificate ("`/ndnfit/alice/sensor/KEY/...`") and the trust anchor certificate ("`/ndnfit/alice/KEY/...`"). Assuming that the consumer application has a pre-configured trust anchor but has no other cached certificates, when it needs to verify the retrieved data, it can fetch all the needed certificates with a single Interest.

VIII. Discussion

A. Comparison of NDN and TCP/IP Security

The differences between NDN security and TCP/IP security originate from the fact that NDN names data whereas IP names locations.

1) Security Properties are Associated with Data: In TCP/IP, the network layer defines the basic communication unit to be the channel between two IP addresses; thus, protocols like IPSec and TLS secure channels (e.g., IP channels or TCP channels). However, (i) when multiple parties are involved, securing channels between every two of them will dramatically increase the communication overheads; (ii) more importantly, even with a protected network channel, a receiver cannot ensure that the transfered data itself is authentic and unaltered – an application cares about the security of its data instead of the security of underlying channels.

By contrast, NDN security allows applications to protect what really needs to be protected – data. NDN's security properties are carried directly with packets, and are thus independent of how the data was retrieved or where it came from.

2) Establishing Trust using Name Semantics: Security technology lacks the tools for effectively reasoning about trust. For instance, in current network security protocols (e.g., HTTPS or QUIC), a common practice is to accept a signature if it has been (in)directly signed by a trusted CA. However, this shows that signature verification alone is not enough to establish trust [10].

In NDN, trust policies are expressed explicitly by using name semantics in a systematic way, allowing applications to reason about security rather than blindly verifying signatures. Moreover, naming conventions can facilitate key distribution, thus improving system usability.

B. Remaining Challenges and Ongoing Work

The development of the NDN architecture has guided the creation of a new network security framework and, at the same time, brought new challenges [11]. For example, stateful forwarding introduces the PIT, which may increase the attack surface [12]. In regard to privacy [13], on the one hand, normal Interest packets [2] fetch Data packets by name only, without disclosing the consumer's information; on the other hand, data names and signatures may disclose a producer's identity if they are not properly protected.

The NDN community is actively working on these challenges. Some ongoing work includes: (i) investigating NDN's DDoS resistance, (ii) supporting multiple certificate chains in Data verification, and (iii) mitigating privacy issues caused by Data names and Data signatures.

IX. Conclusion

In [14], we argued that, by naming data and securing it directly, NDN offered intrinsic advantages for securing network communications. Evidence from our seven-year effort to develop NDN security solutions suggests that this is indeed true. NDN enables data signing at a fine granularity in support of the least-privilege principle; secured Data packets (including certificates and trust schemas) can be easily fetched from anywhere. Furthermore, we learned that one can establish well-defined naming conventions to systematically define trust policies using schemas and design name-based access control via encryption. We also learned, the hard way, the importance of automating security operations instead of leaving the burden to application developers (who would simply aim to make the application work first by leaving security out).

Consequently, NDN secures network communications in a more resilient, intuitive, and less fragmented manner than the existing solutions implemented in TCP/IP networks. The development process of the NDN security model has convinced us that the right building block of the network architecture—named data, offers the key enabler to developing effective network security solutions.

[2]not including signed Interest packets

Acknowledgment

This work is partially supported by the National Science Foundation under awards CNS-1345142, CNS-1345318, CNS-1629009, and CNS-1629922.

References

[1] L. Zhang, A. Afanasyev *et al.*, "Named Data Networking," *ACM SIGCOMM Computer Communication Review*, 2014.

[2] R. L. Rivest and B. Lampson, "Sdsi-a simple distributed security infrastructure." Crypto, 1996.

[3] H. Zhang, Z. Wang *et al.*, "Sharing mhealth data via named data networking." in *ICN*, 2016, pp. 142–147.

[4] Z. Zhang, A. Afanasyev, and L. Zhang, "Ndncert: universal usable trust management for ndn," in *Proceedings of the 4th ACM Conference on Information-Centric Networking*. ACM, 2017, pp. 178–179.

[5] Y. Yu, A. Afanasyev *et al.*, "Schematizing trust in named data networking," in *Proceedings of the 2nd International Conference on Information-Centric Networking*. ACM, 2015, pp. 177–186.

[6] M. Mosko, E. Uzun, and C. A. Wood, "Mobile sessions in content-centric networks," in *IFIP Networking*, 2017.

[7] Z. Zhang, Y. Yu, A. Afanasyev, J. Burke, and L. Zhang, "Nac: name-based access control in named data networking," in *Proceedings of the 4th ACM Conference on Information-Centric Networking*. ACM, 2017, pp. 186–187.

[8] C. Marxer and C. Tschudin, "Schematized access control for data cubes and trees," in *Proc. of ACM Conference on Information-Centric Networking*, 2017.

[9] M. Mittal, A. Afanasyev, and L. Zhang, "NDN certificate bundle," NDN, Technical Report NDN-0054, 2017.

[10] C. Cimpanu. (2017) 14766 let's encrypt ssl certificates issued to paypal phishing sites. [Online]. Available: https://www.bleepingcomputer.com/news/security/14-766-lets-encrypt-ssl-certificates-issued-to-paypal-phishing-sites/

[11] R. Tourani, S. Misra, T. Mick, and G. Panwar, "Security, privacy, and access control in information-centric networking: A survey," *IEEE Communications Surveys & Tutorials*, 2017.

[12] C. Ghali, G. Tsudik, E. Uzun, and C. A. Wood, "Closing the floodgate with stateless content-centric networking," in *Computer Communication and Networks (ICCCN), 2017 26th International Conference on*. IEEE, 2017, pp. 1–10.

[13] C. Ghali, G. Tsudik, and C. A. Wood, "When encryption is not enough: privacy attacks in content-centric networking," in *Proceedings of the 4th ACM Conference on Information-Centric Networking*. ACM, 2017, pp. 1–10.

[14] L. Zhang *et al.*, "Named data networking (NDN) project," NDN Project, Tech. Rep. NDN-0001, October 2010.