Revision 1: March 18, 2018

Security Support in Named Data Networking

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

***Abstract*—This technical report presents an overview of the security support in the Named Data Networking (NDN) architecture that has been developed over the recent years. NDN changes the communication model from IP's delivery of packets between hosts identified by IP addresses to the retrieval of named and secured data packets. Consequently NDN fundamentally changes the approach to securing communications. Making named data the centerpiece of the architecture leads to a new security framework which: (i) secures the data directly, and (ii) uses name semantics for applications to reason about security. In this paper we introduce NDN's approach to security bootstrapping, data authentication, integrity, confidentiality, and availability.**

Note that this report is still in preliminary stage. We welcome all comments, and we plan to post an updated version in the near future.

***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 names. Naming enables NDN to secure *data* directly at the network layer by making every data packet verifiable and, if necessary, confidential.

In this report, we give an overview of the security supporting mechanisms in NDN, introducing security approaches used in NDN with example prototype realizations and showing how all the components of the framework work together. We assume that the readers have some basic knowledge of cryptographic security, but may not be familiar with the NDN architecture. Therefore, we organize this report in the following way:

- Section II introduces the basic notions of NDN, and an example application that will be used throughout the report to help illustrate the NDN security mechanisms.
- Section III provides an overview of the NDN security supporting mechanisms and building blocks.
- Section IV introduces the security bootstrapping process in NDN.
- In Sections V, VI, and VII, we explain how security in NDN provides data authentication, integrity, confidentiality, and availability, respectively.

Throughout this report, we aim to illustrate how NDN enables data to stay secured independent of the communication channel, and how it allows applications to validate received Data packets independent of how or from where the data is fetched. Moreover, we illustrate how applications can utilize name semantics to reason about trust and security, instead of blindly relying on third-party certificate services.

In Section VIII, we discuss the basic differences between network security solutions in TCP/IP and NDN, explaining how different network architectures lead to different security solutions. We also identify remaining NDN security challenges (Section IX). We hope that this report can serve as a roadmap to the 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 practices.

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

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

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

II. Background

NDN security is based on the public key cryptography, which has been widely used to provide solutions for information authentication, integrity, and confidentiality. In public key cryptography, a user generates digital signatures and encrypts content [1] using its public and private key pair, and the key pair is bound with the its identity through a certificate issued by a certificate authority (CA). Utilizing public key cryptography requires NDN to address three challenges.

Establishing trust anchor(s) Only with proper trust anchors, participants can authenticate CAs and verify other entities' signatures by backtracking and verifying all the certificates along the certificate chain. Thus, how to install trust anchors in a secure way during security bootstrapping is of vital importance.

Providing effective solutions for trust management Trust policies are needed to validate all received data; a participant must know which key(s) is legitimate to sign or encrypt which piece of data. Effective solutions must enable applications to express their trust policies, and to execute the policies automatically.

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

In the next subsection, we establish the basic NDN terminology. The rest of the paper will show how trust anchors are installed and how key/trust management is handled in NDN.

A. Named Data Networking (NDN)

The proposed Named Data Networking (NDN) architecture [1] makes application data the narrow waist of the networking stack. Applications create **Data** packets, name them using structured names, while NDN forwarders directly use these names for forwarding at the network layer. Consumers

[1] usually used together with symmetric-key algorithms

request **Data** packets by sending **Interest** packets that carry names of the desired content. Importantly, data producers secure generated content at the creation time— cryptographically signing Data packets and, when needed, encrypting them.

Network communication participants (called **entities** in this paper) in NDN are supposed to possess both names and cryptographic keys, and NDN certificate serves to glue them together; an NDN certificate certifies an entity's ownership of a name along with its key. We call an certified name an **identity** of an entity. All entities in NDN network (e.g., a user, a node, or an application) should have at least one identity and they can have many identities and corresponding cryptographic keys. Figure 1 shows an example of two entities in NDN. Both Alice and Bob their digital keys, trust policies, and pre-configured trust anchors. Two CAs directly or indirectly issue certificates to two entities respectively.

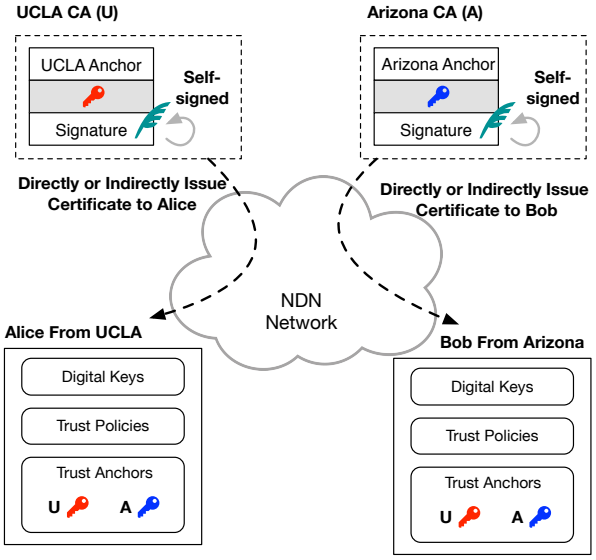


Fig. 1. Entities and Trust Anchors in NDN

NDN utilizes a stateful packet forwarding mechanism [2], where neither Interest nor Data packets carry "source" or "destination" addresses. Each router forwards Interests based on their names, recording the interfaces from which Interests were received and to which they were forwarded in Pending Interest Tables (PIT). Based on that, each matched Data packet will follow the reverse path of the corresponding Interest back to the consumer by satisfying the corresponding PIT entry. This stateful forwarding creates a closed feedback loop, enabling intelligent forwarding decisions for Interests based on the observed performance. Routers, if capable, will also cache the retrieved Data packets to serve future requests with the same name.

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 mechanisms provided by NDN security. In a specific use case of NDNFit, a data owner, "Alice", has a laptop and a mobile phone that run two application instances. Specifically, Alice uses

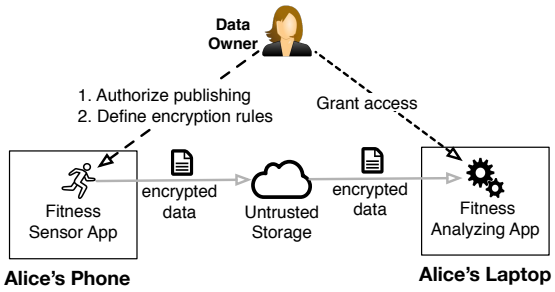


Fig. 2. NDNFit application workflow

- a sensor app (producer) running on her mobile phone to collect her everyday time-location data;
- a public in-network storage unit to store her data; the unit could be a router or a device providing storage service; and
- an analyzing app (consumer) running on her laptop to provide inferred insights and visualized results.

Figure 2 illustrates the NDNFit application workflow. In our example, Alice has identity "`/ndnfit/alice`", and she delegate sub-namespaces "`/ndnfit/alice/sensor1`" and "`/ndnfit/alice/analyzer1`" as identities for sensor application and analyzing application respectively.

The security objective of NDNFit is to allow Alice to directly control applications' access to her data. In our case, Alice grants privilege to the analyzing app to access her private fitness data produced by the sensor app. At the same time, the system should ensure the authentication, integrity, and confidentiality of Alice's private data. The rest of the paper will show how mechanisms in NDN helps NDNFit achieve these objectives.

III. An Overview of Security in NDN

NDN takes a different approach to trust anchor establishment from today's practice. Instead of starting from several commercial CAs (e.g., TLS certificates) or single trust anchor (e.g., DNSSEC), NDN takes the strategy of Simple Distributed Security Infrastructure (SDSI) [4] to establish trust anchors; that is, each network system (network of an organization, a smarthome, etc.) has their local trust anchors. Moreover, NDN utilizes semantics of namespace for trust management (Section V-A) and key management (Section VI-A).

A. Building Blocks of Security in NDN

NDN security utilizes public key cryptography and relies on the use of **digital keys**. Besides keys, NDN also uses the following building blocks, namely, trust policy and NDN certificate.

Trust Policy Trust policies are defined by applications to determine whether a packet or an identity is trustworthy or not. Given the fact that applications name Data packets (including certificates) in a structured and meaningful way, consumers can restrict the name format of a valid packet and trustworthy name relationships between a packet and its signing key. These name-based rules are trust policies in NDN. See more in Section V-A.

NDN Certificate In NDN, every entity that produces data needs to obtain an NDN certificate to prove the ownership of its namespace and cryptographic materials (e.g., public key). Importantly, as the root of trust, *trust anchors* are also represented by certificates. Regarding the format, NDN certificate is a Data packet that carries public key information and can be fetched by normal Interest packets. Certificate name follows naming convention "`/<prefix>/KEY/<key-id>/<issuer-info>/<cert-version>`", where the "prefix" represents an identity and the components after "KEY" are key id, issuer information, and certificate version.

B. Design Considerations of NDN Security

Security in NDN is based on named data, leading to different design considerations of security framework.

- NDN security should be able to support different trust models for different systems.
- Name and naming convention explicitly conveys desired information and can facilitate trust and key management in NDN.
- To work with NDN's content-centric nature, security properties (data authentication, integrity, and confidentiality) should stay with the data regardless of its location.

IV. Security Bootstrapping in NDN

Security bootstrapping in NDN is the process for entities to learn trust anchors and obtain certificates.

- From an application's perspective, to generate Data packets with legitimate names and verifiable signatures, an application (producer) needs to obtain a name and a certificate for the name. Furthermore, to obtain a certificate from a CA, an entity needs to trust the CA first.
- Not only application, a user or a node is also expected to get a certificate, serving as a CA for local devices and applications. To be more specific, after a user obtains an NDN certificate, the user can issue delegated certificates to authorised devices or applications. Similarly, after a node obtains a certificate, the node is able to delegate certificates to authorised applications.

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 CA, e.g., cloud-based applications, while a distributed application, e.g., p2p applications, may obtain certificates from its user or host node.

The prerequisite of security bootstrapping in NDN is name assignment; that is, a node or an application instance has been assigned a name already or has means to get a name. How to obain a name from a namepace should be determined by the owner of the namespace. For instance, a user "Alice" with identity "`/alice`" can name her phone to be "`/alice/my-phone`" and Google may name an application running on a device to be "`/google/user-id/device-id/application-id`".

A. General Steps for Security Bootstrapping

An entity needs trust anchors to distinguish authentic entities; at least, it should trust the CA which will issue a certificate to this entity later. The trust anchors are expected to either be pre-configured or securely obtained by other means, e.g., out-of-band process. Following SDSI, different system may have their own trust anchors and nodes in these systems can have their own ways to obtain trust anchors. Thus, the process of obtaining trust anchors can be localized or distributed in NDN.

With trust anchors, an entity could apply for a certificate from a trusted CA, which can be done either manually or through automated processes, e.g., NDN certificate management system (NDNCERT) [5]. NDNCERT provides tools and library interfaces for entities to automatically apply for certificates. Also, a CA can perform certificate renewal and revocation processes via NDNCERT without manual operations.

B. Security Bootstrapping for NDNFit

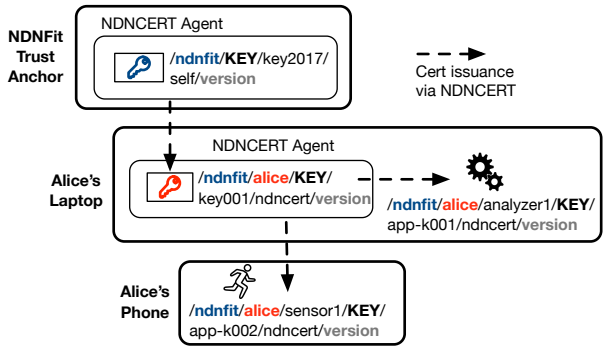


Fig. 3. NDNCERT automate certificate issuance in NDN

NDNFit uses NDNCERT for certificate issuance. In a concrete example of NDNFit bootstrapping, we assume a user "Alice" is the legit owner of name "`/ndnfit/alice`" and owns a certificate "`/ndnfit/alice/KEY/key001/ndncert/version`". Figure 3 shows how NDNCERT helps analyzing app and sensor app to obtain certificates automatically.

- Alice's certificate resides in NDNCERT daemon (called an agent) running in a laptop; an agent plays thes role of CA.
- In application security bootstrapping phase, since NDNFit is a localized system and there is no centralized CA, both sensor and analyzing apps do not have pre-installed trust anchors. By NDNFit's design, two applications will apply certificates from a user, e.g., "`/ndnfit/alice`", through sending requests to the laptop's NDNCERT agent.
- Alice will make use of the agent to verify two apps, by using customized out-of-band challenges, and then issue two certificates "`/ndnfit/alice/sensor1/KEY/app-k002/ndncert/version`" and "`/ndnfit/alice/analyzer1/KEY/app-k001/ndncert/version`".

V. Authentication and Integrity

NDN requires producers to sign every individual Data packet, enabling consumers to verify each incoming Data's signature, hence ensuring data authentication and integrity. More importantly, NDN's rich name semantics enables 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 limits the power of each signing key and ensures each trustworthy packet is signed by a legitimate key, providing data authentication in a fine granularity.

Moreover, an entity is able to sign Interest packets when Interests should be authentic. In IoT scenario, for example, when receiving an Interest packet containing a command, a smarthome device needs to authenticate the sender of the Interest before executing the command. For this purpose, a controller can send a signed Interest to command IoT devices. In NDN, Interest signature validation process is the same as that of Data packets.

The authentication and integrity of the incoming Data packets (including certificates) are determined by a combination of two main factors– validation by name-based trust policies and signature verification.

Validation by Name-based Trust Polices Structured naming convention of Data packets and keys provides explicit and meaningful contexts for applications, enabling NDN applications to define rules that only accept packets with desired format of names and name relationships between packet and 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 the rules.

Signature Verification To verify data signatures, consumers retrieve certificates of the corresponding producers, which are identified by the key names in the dedicated section of the Data packets. The certificate will recursively point to its CA and finally arrives 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 Name-based Trust Policies by Trust Schemas

NDN's "Trust Schema" [6] is used for an application to present its name-based trust policies. Specifically, trust schema makes use of NDN's naming conventions to enable systematic descriptions of the trust policies: (1) how Data packet name is expected to be structured (2) how the packet signing key name is expected to be structured and (3) how Data packet name is expected to be related to the signing key name (4) which trust anchors are accepted

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 shown in Figure 4, in NDNFit, two users "Alice" ("`/ndnfit/alice`") and "Bob" ("`/ndnfit/bob`") are signed by the same anchor certificate "`/ndnfit/KEY/key2007/...`". Both Alice and Bob produce data packets under their own prefix, namely Data "`/ndnfit/alice/data`" and "`/ndnfit/bob/data`". As shown, there are two trust schemas. Schema "rule 1" accepts Data packets whose (1) name prefix is "`/ndnfit/alice`" (2) signing key name prefix is "`/ndnfit/alice/KEY`" (3) certificate chain ends with trust anchor "`/ndnfit/alice`". Accordingly, only packets signed by Alice and strictly under Alice's prefix are accepted. In contrast, "rule 2" has a loose requirement; all packets with 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 and Bob.

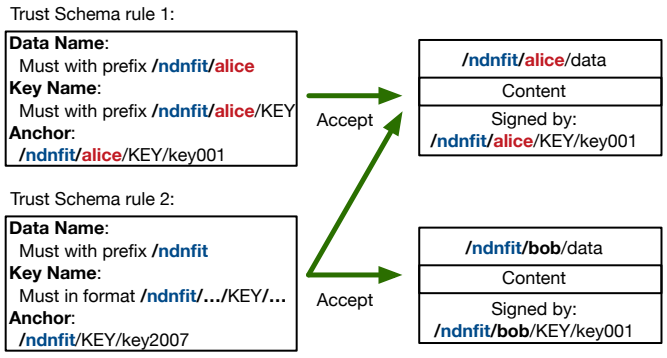


Fig. 4. An example of Trust Schema

VI. Data Confidentiality

When data confidentiality and access control are implemented via encryption, it is necessary to have a usable key distribution system to ensure that the involved entity can figure out and fetch the relevant encryption and decryption keys. In NDN, encryption-based approaches to confidentiality are designed and implemented for different scenarios.

- For active point-to-point sessions, key exchange protocols such as Diffie-Hellman [7] can derive encryption keys for the session and both sides are well aware of the key information.
- For ad hoc and intermittent environments, and when sharing data with multiple parties, Diffie-Hellman may not be feasible or efficient. The owner of the data needs to deliver decryption keys to authorized entities. In NDN, since Data names are structured and convey rich semantics, NDN team is exploring approaches that leverage systematic naming conventions to notice consumers how to fetch corresponding decryption keys, hence automating the key distribution process and improving the protocols' usability.

A. Name-based Access Control

Named-based Access Control (NAC) [8] with its variations such as NAC-ABE, schematized access control [9], and other protocols are being designed and implemented. We take NDNFit which uses NAC to achieve access control as an example; in such systems, a "data owner" (e.g., Alice is the data owner for all Data packets under "`/ndnfit/alice`") determines who under which conditions can access the confidential data. In NDNFit, each encryption key name

will be explicitly appended to the name of the corresponding Data packet. For instance, a Data packet produced by sensor app has name "`/ndnfit/alice/sensor1/data/ENCRYPTED-BY /ndnfit/alice/E-KEY/sensor1`", where the components after "`ENCRYPTED-BY`" is the encryption key name.

In our NDNFit example, analyzing app "analyzer1" "`/ndnfit/alice/analyzer1`" is authorized by Alice to access sensor's data under prefix "`/ndnfit/alice/sensor1`". A simplified data production and encryption process is illustrated as follows and in Figure 5.

Key Generation Alice will first generate a pair of key "E-KEY", "D-KEY" for encryption and decryption. She then produces two Data packets carrying "E-KEY" in plaintext and "D-KEY" encrypted by "analyzer1"'s public key. "E-KEY" packet name is in the format of "`/ndnfit/alice/E-KEY/sensor1`" while "D-KEY" packet name follows the format "`/ndnfit/alice/D-KEY /analyzer/ENCRYPTED-BY/ndnfit/alice/analyzer1`".

Data Production When producing data, the sensor app "sensor1" first generates a symmetric key for content encryption. Then it fetches "E-KEY" and encrypt the symmetric key and packs the encrypted symmetric key into a Data whose name is "`/ndnfit/alice/sensor1/data /ENCRYPTED-BY/ndnfit/alice/E-KEY/sensor1`".

Data Consumption When the analyzing app "analyzer1" 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, "analyzer1" then fetches the corresponding "D-KEY". Notice the fetched "D-KEY" is actually encrypted by "analyzer1"'s own key. By decrypting the content in fetched Data, the application gets "D-KEY" and can finally decrypt the content.

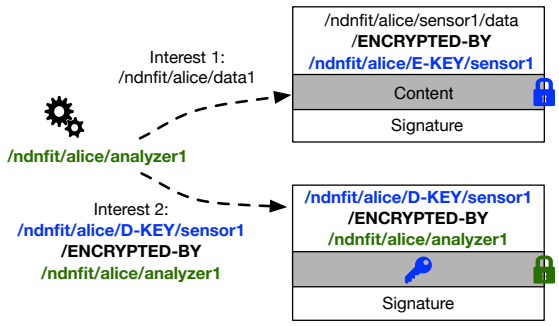


Fig. 5. Name-based Access Control in NDNFit

B. Fine Granularity by Key Naming

By designing the naming convention, NAC enables fine-grained access control. For instance, a "D-KEY" name could be "`/ndnfit/alice/D-KEY/sensor1/position/monday`", hence the key will only used to protect position data produced by "sensor1" on Monday.

VII. Data Availability and Certificate Availability

A. Data Availability by In-network Storage

NDN provides applications with easily-achieved high data availability due to its content distribution nature. 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 systems, no matter these cache services and storage systems are trustworthy or not. All capable forwarders can cache Data packets and popular Data packets are usually cached in the network. Moreover, it's easy for producers to utilize long-term storage services (e.g., managed data repositories). In NDNFit example (Figure 2), the middle box provides cache service and helps to improve the availability of Data produced by "sensor1".

B. Certificate Availability

The fact that certificate is a building block of NDN security makes certificate availability of vital importance. NDN certificates as Data packets also benefits from in-network cache and storage. For instance, in cases when a certificate authority goes down, the cached certificates can still support the normal signature verification for a period of time.

In order to further improve the certificate availability, the NDN certificate bundle [10] has been designed to improve the certificate availability. Certificate bundle enables producers to aggressively collect all certificates in the certificate chain and pack them together as a bundle, hence providing the whole chain of certificates for consumers and avoiding signature verification failures due to unreachable certificates. In our NDNFit example, sensor application "sensor1" produces Data packets and it can prepare all needed certificates as a certificate bundle. Specifically, the bundle will contain application certificate "`/ndnfit/alice/sensor1/KEY/...`", data owner certificate "`/ndnfit/alice/KEY/...`", and NDNFit trust anchor "`/ndnfit/KEY/...`". Assuming the consumer application has pre-configured trust anchor but has no any other cached certificates, when it needs to verify the retrieved data, it can fetch all the needed certificates with a single Interest.

VIII. Discussion

Different architecture decisions lead to different security in TCP/IP and in NDN. Current TCP/IP network names hosts identified by locations, while NDN names data, leading to the different security framework as we discussed in previous sections.

A. Security in Today's Internet

Addresses being the centerpiece, current Internet architecture provides upper layers with no more than IP addresses. Thus, most existing network security protocols, such as IPSec, OpenVPN, TLS, QUIC [11], SSH, etc, secure network communication by protecting the channel. However, Public Key Cryptography binds a public key with a named identity (e.g., an organization, a DNS name, and etc.) instead of an address, causing the incongruence and increasing the complexity when

deploying security. Another fact is that underlying mechanisms of establishing cyberspace trust vary among protocols. While IPSec and OpenVPN rely on manually-configured certificates, TLS, QUIC, and others put their trust in a set of globally-trusted certificate authorities (CAs), and SSH has an option of using "trust on the first use" (TOFU) model.

An exception is DNSSEC [12]; it secures DNS records directly instead of channels. DNS provides naming service (data-centric) and relies on caching, thus protecting channel is not efficient. DNSSEC shows an example of building a unified and global security system for DNS, and how trust is derived within the hierarchy of the domain names. Nevertheless, its security framework is restricted to hierarchical relations only (root's key signs ".com", ".edu", etc., ".edu"'s key signs "ucla.edu", etc.) and applies only to resource record of DNS instead of general data.

B. NDN Security: Comparison with TCP/IP Security

1) Security Stays with Network Data: In TCP/IP network, before a channel is protected by security mechanisms, a fundamental requirement is to ensure the authentication of two sides (IP addresses at least), which is non-trivial. Moreover, when multiple parties are involved, protecting one-to-one channels becomes inefficient; applications usually need to cope with security in application layer by themselves, e.g., web of trust. Importantly, since network layer does not provide a unified interface for security, approaches to network security vary widely among existing protocols, leading to a fragmented security ecosystem and additional configuration overhead for each additional protocol used.

In NDN, security properties come directly with Data packets and are independent from where those packets come or stay. Furthermore, NDN provides a standard packet format for security purpose and Data packets produced by applications are directly used in network layer without extra headers. In this way, all data packets can be verified in the same manner by all applications, forming a unified authentication ecosystem.

2) Reason About Trust using Name Semantics: Security technology lacks the tool for reasoning about trust effectively. For instance, being widely used in current network security protocols (e.g., HTTPS, QUIC), a common practice is to accept a signature if it is eventually signed by a trusted CA.

In NDN, trust relationships are expressed explicitly by the name-based trust policies in a systematic and semantic way. Name semantics enable applications to define their own trust policies.

3) Reduced Cost and Dependencies of Security: Applying security as a patch to the network stack makes network security expensive. (1) It usually leads to additional round trips. For example, compared to pure the TCP connection setup, adding TLS 1.2 session setup involves three more round trips. TLS 1.3 is trying to address the overhead by utilizing the pre-shared key and zero-RTT Data and QUIC applies TLS over UDP to reduce the setup round trips. (2) Current network security cause duplicated workload for communication participants. For instance, assuming the communication is protected by TLS 1.2, even when the requested content is the same, the server needs to encrypt the same piece of content as many times as the number of the clients. (3) Dependencies such as DNS increase the potential attack surface.

In contrast, within NDN, all Data packets are self-describing and protected when created, hence it requires no extra steps for applications to cryptographically secure the channel. When content is encrypted for confidentiality, the Data packet is not bound with a specific consumer address and can be reused by all authorized consumers. As a result, data producers do not have to perform the same task multiple times, making their workload lighter. Importantly, NDN shares the same data chunks with applications, hence when Data packets are secured at the network layer, upper layers are naturally protected without extra dependencies.

4) Improved Privacy of Consumer: In NDN, since normal Interest packets fetch Data packets by name, they do not disclose any information about consumers, while fetched Data packets also contain nothing related to the consumers. Therefore, NDN's retrieval model does not expose consumers' privacy. On the other hand, IP header carries the source address, and malicious ones can dig even more information from the traffic related to that address. Eavesdroppers and Internet Service Providers (ISPs) can easily access sensitive meta-information (e.g., IP addresses with round-trip time).

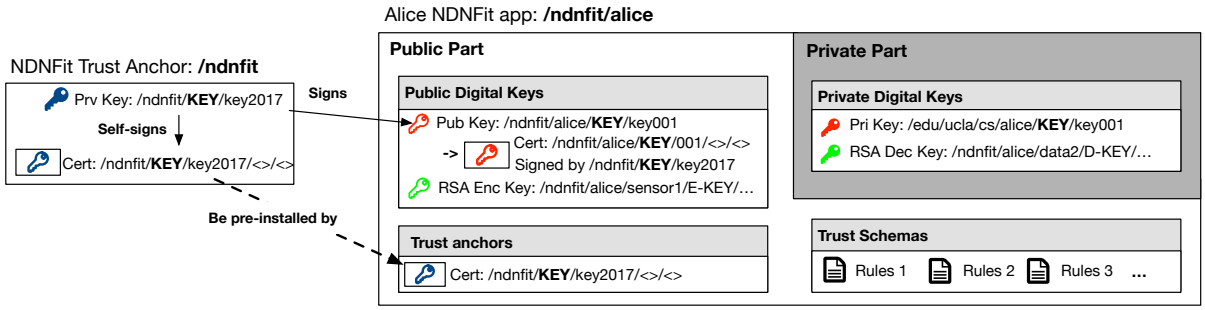
Notice that signed interest is for the purpose of authentication and will reveal consumer's information if the signature information components are not encrypted.

5) Mitigated Denial-of-Service: NDN's communication pattern naturally mitigates Denial-of-Service(DoS) attacks. First, DDoS attacks through Data packets are not possible, since an NDN endpoint will not receive unsolicited Data packets unless it has sent out an Interest requesting this Data. Second, Interest-based DDoS attacks can be mitigated, since at routers/forwarders, Interests with the same name are aggregated, reducing the number of Interests received by producers. NDN potentially allows new types of attacks, such as Interest flooding. However, these attacks can be mitigated using the same core features of NDN forwarding: stateful forwarding. For example, routers can evaluate Interest satisfaction statistics to make decisions on whether to accept, rate-limit, or reject an incoming Interest [13].

IX. Remaining Challenges and Ongoing Work

1) Name Confidentiality: Names contain rich information and may degrade privacy on the network. For instance, the name "`/ndnfit/alice/sensor1/bloodsugar/...`" reveals a good deal of personal information, opening the door for attackers to analyze and exploit private information. Data signatures contain information about signing keys and producers, also disclosing valuable information to attackers. How to prevent a Data name from hurting the confidentiality of a Data packet is a challenge. To safeguard privacy, ongoing efforts like ANDaNA [14] focus on exploring the name encryption mechanisms.

2) Bootstrapping Trust: As mentioned in NDN's security bootstrapping, an entity should learn trust anchors before utilizing tools like NDNCERT to obtain certificates. In today's



Alice's app holds identity /ndnfit/alice and its cryptographic materials. The public part contains public keys (public keys of identities, NAC encryption keys), trust anchors, and trust schemas. The Private part contains private keys (e.g., signing keys of identities, NAC decryption keys, Symmetric keys, Shared Secrets). Specifically, the blue key pair belongs to NDNFit CA, serving as the trust anchor in NDNFit system; the red key pair is Alice's identity key pair which is bound with her name /ndnfit/alice; the green key pair is E-KEY and D-KEY which are used in NAC.

Fig. 6. A Snapshot

TCP/IP networks, a typical example is to establish initial trust through pre-installed certificates in web browsers. However, in NDN, not only an application but also a node or a user needs security bootstrapping. NDN team is currently working on security bootstrapping in different scenarios (e.g., IoT).

3) Multiple Authentication Chains: The same piece of content may be signed with different keys and each digital key may further be endorsed by multiple certificates of different namespaces. In this way, a data producer offers consumers multiple certificate chains for data authentication. However, it is unclear as of yet how multiple separate chains can be associated with one Data packet and how consumers choose a proper chain. [15] has made initial exploration of multiple signature scheme in NDN.

X. Conclusion

A snapshot of an entity's insight of NDN security is shown in Figure 6.

In [16] we argued that, by naming data and securing it directly, NDN offers 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 fine-granularity of data signing in support of the least-privilege principle; secured data packets can be fetched from anywhere; and certificates and trust schemas are also simply named, secured data packets and can be easily fetched from anywhere. Furthermore, we learned that one can establish well-defined naming conventions to systematically define trust rules 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 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 offers the key enabler to developing effective network security solutions.

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] C. Yi, A. Afanasyev, I. Moiseenko, L. Wang, B. Zhang, and L. Zhang, "A case for stateful forwarding plane," *Computer Communications*, vol. 36, no. 7, pp. 779–791, 2013.

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

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

[5] 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.

[6] 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.

[7] M. Mosko, E. Uzun, and C. Wood, "Ccnx key exchange protocol version 1.0," Working Draft, IETF Secretariat, Internet-Draft draft-wood-icnrg-ccnxkeyexchange-02, July 2017, http://www.ietf.org/internet-drafts/draft-wood-icnrg-ccnxkeyexchange-02.txt. [Online]. Available: http://www.ietf.org/internet-drafts/draft-wood-icnrg-ccnxkeyexchange-02.txt

[8] 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.

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

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

[11] A. Langley, A. Riddoch *et al.*, "The quic transport protocol: Design and internet-scale deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*. ACM, 2017, pp. 183–196.

[12] R. Arends, R. Austein, M. Larson, D. Massey, and S. Rose, "Dns security introduction and requirements," Internet Requests for Comments, RFC Editor, RFC 4033, March 2005. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4033.txt

[13] A. Afanasyev, P. Mahadevan, I. Moiseenko, E. Uzun, and L. Zhang, "Interest flooding attack and countermeasures in named data networking," in *2013 IFIP Networking Conference*, May 2013, pp. 1–9.

[14] S. DiBenedetto, P. Gasti, G. Tsudik, and E. Uzun, "Andana: Anonymous named data networking application," *arXiv preprint arXiv:1112.2205*, 2011.

[15] Y. Yu, A. Afanasyev, Z. Zhu, and L. Zhang, "An endorsement-based key management system for decentralized ndn chat application," NDN, Technical Report NDN-0023, 2014.

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