UNIVERSITY OF CALIFORNIA
Los Angeles

# Named, Secured Data:
# A Fundamental Building Block for Secure Networking

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Zhiyi Zhang

2021

ABSTRACT OF THE DISSERTATION

Named, Secured Data:
A Fundamental Building Block for Secure Networking

by

Zhiyi Zhang
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2021
Professor Lixia Zhang, Chair

Securing network communications is one major goal of the Internet. Due to the point-to-point communication model of TCP/IP architecture, at the time when security becomes a necessity, the channel-based security model, represented by Transport Layer Security (TLS), was applied to secure network communication between hosts. However, with the growth of content delivery applications and networks, recent years have seen the mismatch between what application needs, *i.e.*, secured data, and what is provided by the channel-based security model, *i.e.*, secured channels.

A newly proposed architecture, Named Data Networking (NDN), has been developed over the past decade. Departing from TCP/IP's network model, NDN considers named secured data, instead of channels, as the building block, and provides an alternative to today's security model by directly securing data instead of channels. Not relying on the network context, a piece of named secured data can be forwarded, cached, and reused without breaking the security primitives. NDN also uses the stateful forwarding to forward named secured data at the network layer.

Under this background, there is an urge to understand the difference between the two security models and explore how to utilize the new way of dong network security to address today's security issues. To answer these questions, we revisit the key concepts of network security from the application's perspective and compare the two security models. Then, we present our design of a number of security solutions built over NDN's named secured data, including a self-contained smart home control system, a DDoS mitigation mechanism that supports fine-grained traffic throttling, a distributed ledger system for distributed rooftop solar energy system, a multiparty signing and verification tool set, and a secured data prefetching system for vehicular networking. We also describe two security solutions built on to the application level for asynchronous and privacy-preserving single sign-on (SSO) and reliable leaker identification in sensitive data sharing, respectively. While not directly built over NDN because of today's deployment constraints, they follow the notion of the data-centric security model. Through the design discussion of these systems, we confirm the unique advantages of the new security model and demonstrate how the data-centric security model and especially, NDN's named secured data, can be applied to address some challenges that are intractable to the channel-based security model.

The dissertation of Zhiyi Zhang is approved.

Leonard Kleinrock

Songwu Lu

Peter Reiher

Alexander Afanasyev

Lixia Zhang, Committee Chair

University of California, Los Angeles

2021

*I dedicate my dissertation to Jiusong, my amazing wife, who has offered unwavering support in the past five years of my doctorate journey. She encourages me when I am upset, believes in me when I hesitate, and laughs with me when I make progress. Her love and sacrificial care of me and Alan, our child, make it possible for me to finish this journey.*

*I also want to thank my mom and dad, who instilled in me the courage to think big and pursue my dream when I was a child.*

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

# ACKNOWLEDGMENTS

First and foremost, I sincerely thank my esteemed advisor, Prof. Lixia Zhang, for her insightful guidance, patience, and continuous support throughout my Ph.D. study. Her relentless pursuit of better networking systems and her diligent, rigorous, and earnest attitude to scientific research have encouraged me in all the time of my academic research and daily life.

I also want to thank my Ph.D. committee members for their time and valuable feedback on my research works, including Prof. Leonard Kleinrock (UCLA), Prof. Songwu Lu (UCLA), Prof. Peter Reiher (UCLA), and Prof. Alexander Afanasyev (Florida International University).

I am grateful to my colleagues at Internet Research Lab, including Yingdi Yu (Facebook), Wentao Shang (Snapchat), Haitao Zhang (Apple), Yanbiao Li (Chinese Academy of Sciences), Philipp Moll, Zhongda Xia, Xinyu Ma, Tianxiang Li, Tianyuan Yu, Eric Newberry, Yuming Xia (Intel), Yukai Tu (Coinbase), Sichen Song, Tyler Liu, and Yufeng Zhang, and my collaborator at UCLA, including Siva Kesava Reddy Kakarla, Vishrant Vasavada, Arthi Padmanabhan, and Zhaowei Tan. I benefited a lot from insightful and productive research discussions with them.

I would also appreciate all the support and insightful discussion I received from my collaborators from NDN project team and all over the world, including Prof. Beichuan Zhang (University of Arizona), Prof. Lan Wang (University of Memphis), Prof. Alexander Afanasyev (Florida International University), Dr. Craig Lee (Aerospace), Prof. Etienne Riviere (UCLouvain), Dr. Alberto Sonnino (Facebook), Dr. Michal Krol (City University of London), Dr. Junxiao Shi (NIST), Dr. Davide Pesavento (NIST), and others.

At last, I would like to gratefully thank my family for their unconditional love and support throughout my whole life, especially my wife Jiusong Tang for supporting and encouraging me in the past eight years.

# VITA

| | |
|---|---|
| 2012–2016 | Undergraduate majored in Software Engineering, Undergraduate researcher at Database and Information System Lab, Awarded the National Scholarship of China in 2015, Nankai University, Tianjin, China. |
| 2016 | Bachelor of Engineering, Outstanding Graduate Student Award, Nankai University, Tianjin, China. |
| 2016–2021 | Graduate student researcher at Internet Research Lab, Teaching assistant in CS118, CS35L, CS217, Computer Science Department, UCLA, California, US. |
| 2017 | Research and development intern at Alibaba Group, Seattle, Washington, US. |
| 2020 | Development intern at Facebook, Menlo Park, California, US. |

# PUBLICATIONS

Zhiyi Zhang, Michał Król, Alberto Sonnino, Lixia Zhang, Etienne Riviere, *EL PASSO: Efficient and Lightweight Privacy-preserving Single Sign On.* In Proceedings of Privacy Enhancing Technologies Symposium (PETS 2021)

Zhiyi Zhang, Tianxiang Li, John Dellaverson, Lixia Zhang, *RapidVFetch: Rapid Downloading of Named Data via Distributed V2V Communication.* In Proceedings of IEEE Vehicular Networking Conference (IEEE VNC 2019)

Yanbiao Li, Zhiyi Zhang, Xin Wang, Edward Lu, Dafang Zhang, Lixia Zhang, *A Secure Sign-On Protocol for Smart Homes over Named Data Networking.* IEEE Communications Magazine (IF=11.052), 2019

Craig Lee, Zhiyi Zhang, Yukai Tu, Alex Afanasyev, Lixia Zhang, *Supporting Virtual Organizations Using Attribute-Based Encryption in Named Data Networking.* In Proceedings of IEEE International Conference on Collaboration and Internet Computing (IEEE CIC 2018)

Zhiyi Zhang, Yingdi Yu, Sanjeev Kaushik Ramani, Alex Afanasyev, and Lixia Zhang, *NAC: Automating Access Control via Named Data.* In Proceedings of IEEE Military Communications Conference (IEEE MILCOM, 2018)

Zhiyi Zhang, H Zhang, E Newberry, S Mastorakis, Y Li, A Afanasyev, L Zhang, *An Overview of Security Support in Named Data Networking.* IEEE Communications Magazine (IF=11.052), 2018

H Zhang, Y Li, Zhiyi Zhang, A Afanasyev, Lixia Zhang, *NDN Host Model.* ACM SIGCOMM Computer Communication Review, 2018

Yu, Yingdi, Alexander Afanasyev, Jan Seedorf, Zhiyi Zhang, and Lixia Zhang, *NDN DeLorean: An authentication system for data archives in named data networking.* In Proceedings of ACM Conference on Information-Centric Networking (ACM ICN, 2017)

Tianyuan Yu, Zhiyi Zhang, Xinyu Ma, Philipp Moll, Lixia Zhang, *A Pub/Sub API for NDN-Lite with Built-in Security.* Named Data Networking Technical Report 0071, 2021

Li, Yanbiao, Alexander Afanasyev, Junxiao Shi, Haitao Zhang, Zhiyi Zhang, Tianxiang

Li, Edward Lu, Beichuan Zhang, Lan Wang, and Lixia Zhang, *NDN Automatic Prefix Propagation.* Named Data Networking Technical Report 0045, 2018

Zhiyi Zhang, Yingdi Yu, Alex Afanasyev, and Lixia Zhang, *NDN Certificate Management Protocol (NDNCERT).* Named Data Networking Technical Report 0054, 2017

# CHAPTER 1

# Introduction

## 1.1 The Development of Network Security

The requirement of network security came along as far back as 1988 when the Morris worm [Orm03] hit more than six thousand computers on the Internet. After that, people start to realize the growing threats in computer networks and network security becomes a common requirement.

The essential abstraction of the network, as indicated by TCP/IP, centers around host identifiers and consequentially, the model of communication is based on point-to-point channels between two hosts. At the time when TCP/IP was designed back in the 1970s, network security is not considered a big concern yet and thus TCP/IP protocols do not have countermeasures to network attacks. In the early 1990s, due to the risk of using unprotected TCP/IP, applications like e-commerce requires network security solutions. As a result of need, a number of supplementary mechanisms such as IPSec [FK11], SSL [FKK11], and its successor TLS [Res18] have been patched to today's network stack. Since TCP/IP is based on the channel established by the host-to-host connection, the security added by these solutions is also bound to channels. That is, the security of data relies on setting up a synchronous channel to a known host, and the security primitives are only applied to data being transferred in a channel.

In 2006, Van Jacobson gave the talk about Content-centric Networking [JST09], in which a new abstraction of networking is proposed – the network centers around the data name,

and the communication model becomes to fetch named data interested by the applications. The idea was later followed by different projects under the umbrella of Information-centric networking (ICN) in Europe and Named Data Networking (NDN) [ZAB14] sponsored by NSF in the U.S. A new network security model called data-centric security was also proposed together to work with the new network model. The main idea is the use of *named secured data*. Being named, the data can be located and fetched by the name rather than the IP address of the host where the data originates, allowing data to be fetched from any host who caches it. Being secured, the security protection of data is self-containing and independent of where the data is fetched or stored in the network. A typical example of a piece of named secured data is composed of a channel-independent data name, the data payload, and a digital signature covering them generated by the data producer. As such, anyone who knows the signer (*i.e.* signer's public key) can authenticate the data regardless no matter if the content is in transmission or cached in some storage systems.

## 1.2 Two Ways of Providing Network Security

Under this background, we first want to understand the difference between the two models with regard to the application's network security requirements.

The ultimate goal of network and network security is to serve applications. This can be confirmed by the network layering model (Figure 1.1) where the application layer is on top of the network layer. To be more specific, network and network security need to help applications to obtain data that can meet their security requirements, including data integrity, authenticity, confidentiality and etc.

**Channel-based security** Channel-based security model works well in the early days of the Internet as clients directly connect to the server over the network infrastructure. This is because channel-based security is sufficient to satisfy application's needs as talking to an

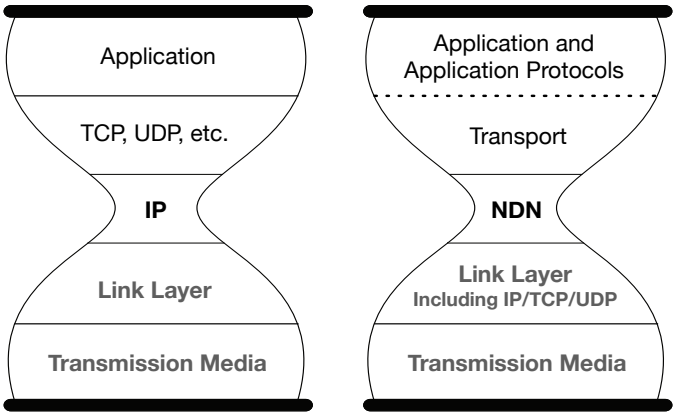



Figure 1.1: TCP/IP and NDN network layering models

known host leads to getting trustworthy data. For example, if the client can ensure that a piece of news is downloaded from the ABC News' server (*i.e.*, the right IP address) through TLS, the client can believe the data is truly from ABC News with the security properties provided by TLS.

However, there are two intrinsic deficiencies in the channel-based security model.

- First, channel-based security was designed to work with synchronous channels over the Internet infrastructure. However, Internet connectivity has changed significantly in the past twenty years. For example, many application services operate over non-infrastructure networks like mobile ad-hoc networks, Internet of Things (IoT), vehicular networks, and so on, and many direct client-server connections no longer exist.

- Second, while what applications care the most is "what", *i.e.*, secured data, the channel-based security answers the question of "who", *i.e.*, setting up a secured channel to a specific host, and thus there is a cognitive mismatch.

The two deficiencies can show up tangled together in today's network systems. A piece of evidence is that, with the growth of content delivery applications, the Content Delivery Network (CDN) services have been widely used to speed up the content delivery by keeping data copies close to the clients (which also confirms our early statement that applications

care about data). As such, the direct synchronous connection between the client and the server has been broken indeed. In addition, there is also a cognitive mismatch because what clients want is the data from content providers but the channels are between clients and CDN providers. Therefore, to maintain the channel-based security while enjoying the benefit of CDN services, today's CDN provider needs to "impersonate" the content provider by obtaining the latter's identity, usually in the form of cryptographic keys that supposed to be known to the content providers only. In this way, the clients can connect to a CDN edge server while authenticating the CDN as the content provider so that the user can trust the data downloaded from the TLS/HTTPS channel.

**Data-centric security** In contrast, directly securing data instead of channels, the data-centric security model cognitively matches what applications need. In addition, since the data-centric security model no longer assumes a synchronous channel, it is friendly to networks that require asynchronous communication, including wireless networks, mobility networks, delay tolerant networks, and so on.

For example, when using NDN, the client can directly verify the data is released by its content provider regardless of whether the data is fetched from a CDN service provider or directly from the content provider. Therefore, there is no need to force the data directly downloaded from the content provider's IP in real time or to let CDN providers compromise the content provider's key.

The concept of data-centric security has also been observed in new trends in web development, confirming applications' need of securing "what" instead of "where". For example, the Web Package [Yas21] mechanism aims to let clients use content when they are offline and in other situations where there does not exist a direct connection to the server where the content is generated. To ensure the security of the content, signed HTTP exchange [Yas20] has been proposed to make an HTTP file just like a piece of named secured data in NDN, whose security is independent of connections. Other works sharing the similar high-level

idea include IPFS [Ben14], Verifiable Credential [SLC19], etc. However, due to the currently deployed TCP/IP architecture, these new mechanisms can only be implemented purely at the application layer, orthogonal or even conflicting to the underlying channel-based security. For example, additional complexity is needed to map the secured objects to legacy IP addresses: IPFS applies the Distributed Hash Table (DHT) mechanism to convert an content hash based IPFS content identifier to IP hosts.

## 1.3 Design Security Solutions with Named Secured Data

Through the analysis of the two security models, we realize the data-centric security model, especially NDN's named secured data, allows to address security issues that may be non-trivial to the current practice of network and network security. For this purpose, we apply NDN's named secured data and designed a number of security solutions to address several real-world challenges, ranging from DDoS mitigation to secure communication in smart home networks and vehicular networks.

- We designed Sovereign [ZGM20] to build autonomous smart home systems with NDN over local broadcast media. This is big departure from today's best practice where the home control dependents on the external cloud and consequently the user's privacy like daily operations are also disclosed to the cloud. In Sovereign, trust is built at the local home instead of the cloud. Utilizing named secured data, resources at home can be accessed with naming conventions without a centralized message broker and are secured without the need of setting up secured channels among home participants. In addition, Sovereign utilizes data names for fine-grained trust management and access control.

- We developed FITT [ZVO19] to mitigate the current network flooding attacks. To work with named secured data, NDN uses a stateful forwarding plane. By utilizing the in-network state, FITT is able to accurately track the real attackers and propagate

the filtering rules to network nodes near the attackers. In addition, with named traffic, FITT can perform fine-grained traffic throttling without bothering benign traffic flows. Importantly, FITT is incrementally deployable with incentives to stakeholders.

- We proposed DLedger [ZVM19] as a general logging component that is immutable, high-performance, and distributed. DLedger is built over a peer-to-peer NDN network, a synchronized data structure based on Direct Acyclic Graph (DAG) of blocks with hash, and a set of security conventions to ensure the robustness of the system. Compared with conventional logging systems and blockchain systems, DLedger ensures immutability while removes the single head of line blocking in single-chain systems.

- We developed RapidVFetch [ZLD19], an data fetching system to facilitate data downloading in vehicular networking. Since it is the requesting vehicle that best knows its future locations and desired content, RapidVFetch lets each vehicle, called a *requester*, express their needs by Interest packets which are small in size, and solicit help from other vehicles, called *forwarding vehicles*, through V2V communication. The forwarding vehicles forward these Interest packets to vehicles at future locations and express them to the RSU, where the data will be prefetched and cached. With named secured data, the desired data can be expressed accurately and the prefetched data can be cached and consumed securely.

- We also designed NDN-MPS [], a multiparty authentication and authorization for distributed energy resources control in the smart grid. Since multiparty signing indicates a trust model that is very different from the conventional producer-consumer trust model in NDN. Therefore, NDN-MPS extends the trust schema to support complex trust policies where multiple signers are involved. With named secured data, multiparty authentication and authorization can be implemented coherently with the network design instead of as a separated piece realized purely at the application layer.

Through the design discussions, we demonstrate a new way of providing network security

departing from today's practice. We show how data-centric security and especially, named secured data, can be applied to address security challenges that are intractable to the channel-based security model, confirming the benefits of the new network security model.

## 1.4 Bring Secured Data to Today's Network Systems

We also designed two systems that are not directly built over NDN but still follow the high-level idea of providing data-centric security to enhance security in the current Internet.

- We developed EL PASSO [ZKS21], a privacy-preserving and asynchronous Single Sign-on (SSO) system. Instead of relying on setting up a synchronous secured channel to the trusted third party (IdP) for each authentication operation, a client can obtain a piece of secured data, *i.e.*, the credential, from the IdP. Furthermore, in EL PASSO, such a credential allows the client to authenticate herself to services in an anonymous way with selective attribute disclosure.

- We designed ReLiShare [ZXG20], a leaker identification system for sensitive data sharing. Instead of using the raw dataset in the sharing process, ReLiShare lets the shared dataset itself carries sufficient information for future leaker identification in case of a data leakage event. As such, the security property for leaker identification still stays with the dataset even after the data sharing is finished, *i.e.*, the channel between the data owner and receiver is ended. In addition, ReLiShare is able to immutably record each sharing process for non-repudiation and identify a guilty data owner (*e.g.*, who leaks data by accident after the sharing), non-colluding guilty receivers, and colluding receivers.

Limited by the deployed architecture and security, the secured data in these works cannot be named for networking. As a result, additional complexity and compromise of two security models are needed for them to fit today's Internet. Note that the statement also applies to

the secured data in other data-centric security solutions in today's Internet, for example, the signed HTTP object in the Web Package or the Verifiable Credential object. In comparison, NDN's way of named secured data is to remove the discrepancy by providing a coherent data-centric network-to-application solution suite.

## 1.5 Outline

In the rest of the dissertation, we first revisit the network security and investigate the difference between the existing channel-based security and data-centric security in Chapter 2. Then in Chapter 3, we systematically describe how NDN provides data-centric security with named secured data and the stateful forwarding. We then show how to utilize NDN's data-centric security support to build security solutions in Chapter 4 and show how to apply the idea of data-centric security to network systems in today's Internet in Chapter 5. In the end, we conclude the dissertation and talk about future works in Chapter 6

# CHAPTER 2

# A Perspective of Network Security

## 2.1 Network Security from Application's Perspective

To help understand the difference between channel-based security and data-centric security, it is important to first make clear the goal of network security. For this purpose, we collect several existing definitions of network security from different sources.

As defined in ISO/IEC 27033-1:2015 [ISO15]:

> "Network security applies to the security of devices, security of management activities related to the devices, applications/services, and end-users, in addition to security of the information being transferred across the communication links. It is relevant to anyone involved in owning, operating or using a network."

As defined in Wikipedia [Wik21b]:

> "Network security is a set of rules and configurations designed to protect the integrity, confidentiality and accessibility of computer networks and data using both software and hardware technologies."

As defined by SANS Institute [SAN21]:

> Network security is the process of taking preventative measures to protect the underlying networking infrastructure from unauthorized access, misuse, malfunction, modification, destruction or improper disclosure. Implementing these mea-

sures allows computers, users and programs to perform their permitted critical functions within a secure environment.

As shown, existing definitions consider both hardware and software security, and both local device security and communication security. However, since the discussion of channel-based security and data-centric security is less related to the hardware and localhost security, for the sake of simplicity, we assume that the local hardware platform and the localhost environment, which is managed by the operating system, are secure when discussing two security models.

With the assumption, we can see existing definitions actually cover two main aspects.

- The security of the network itself, including the security of the network devices and networking infrastructure.
- The security of communications and the data exchanged over the network. This is to serve the applications, services, and end-users.

Since the network devices and the networking infrastructure are also managed by applications (*e.g.*, the routing process can be considered as an application process, SDN controllers) and application layer protocols (*e.g.*, DNS, DHCP), we can unify the two into one, that is, the network security is to provide applications with secure communications and data exchange.

We want to highlight the importance of applications and understand network security from the application's perspective. This is because the ultimate goal of the network is to serve applications, confirmed by TCP/IP model, the OSI model, the NDN model, and many other network system architectures. Considering the network security is to secure the networking process, eventually, the goal of network security is also to serve applications.

Therefore, to understand network security, we first need to answer one question – what is the application's need? Is the need to establish a secure channel between network participants or to ensure the data sent to and received from the network is secure? The first answer

leads to today's network security solutions which center around channels while the second answer leads to the data-centric security model. We argue the application's need is secure data instead of secure channels. This conclusion can be inferred from several aspects: (i) applications serve human users and human users care about the data instead of the channels; (ii) the goal of establishing secure channels is also to securely send and receive data.

To this end, combining the goal of network security and what application's expectation of secured data, we give the following description of network security. Network security refers to the protection of data transferred in the network to meet application requirements of data availability, integrity, authenticity, confidentiality/access control, privacy considerations, and other security related properties.

In the rest of this chapter, we will discuss the channel-centric security model and data-centric security model, and analyze their difference based on this notion.

## 2.2 Channel-based Security

As stated, channel-based security was a necessity as the result of need, instead of an option, at the time. The consequence is that the channel-based security model becomes the default model for TCP/IP architecture. For example, web security is largely based on the HTTPS protocol which secures the channels between clients and web service providers. With such a secured channel, a client can authenticate the server's possession of the claimed domain name, and both sides of the channel can ensure the authenticity and secrecy of the transferred data. Not only being used in web, TLS, the underlying protocol of HTTPS, has also been widely used in many other network systems, including Simple Mail Transfer Protocol (SMTP), Virtual Private Network (VPN), Onion Routing, Secure Shell (SSH), File Transfer Protocol (FTP), etc.

In the channel-based security model, whether the data is secure depends on establishing the secure channel (*e.g.*, a TLS connection) with an expected host. Importantly, the security

primitives are applied to data only when the data is in the synchronous channel. Once a piece of data leaves the channel, the security properties of the data no longer exist. For example, after a client Alice downloads an HTTP file from ABC News, Alice cannot share this file with another client, Bob, and prove to Bob that the file was generated by ABC News. Therefore, Bob needs to establish a channel to ABC News' server and download the same file again.

Based on these characteristics, we give the following summary of the channel-based security model.

- The security of data relies on setting up a synchronous channel to an authentic host where the data originates.
- The security properties of data exist only in the context of the channel.

When asynchronous data exchange is needed, for example, in mobile ad hoc networks, wireless networks, delay tolerant networks, and networks with intermittent connectivity, applying channel-based security can cause additional and undesired complexity in the system.

**End-to-end Principle is Being Broken** In a relatively simple scenario where clients directly build TCP/IP connections to the servers, the channel-based security model follows the end-to-end principle as the security can be ensured by two end hosts without much help from middle network devices.

However, the end-to-end principle has been gradually broken since the use of middleboxes that break the end-to-end TCP/IP connection between clients and servers, especially when the middleboxes are provided by a different party from the service provider, *e.g.*, CDN service providers. This is because the security is no longer added or verified by two end hosts, instead, it is the middleboxes who take the responsibility of the client (*e.g.*, forward proxies) or the server (*e.g.*, reverse proxies) in the communication. As a result, there is a

discrepancy. For example, when CDN is used, the clients perceive that the data is securely downloaded from the content provider, but in fact, it is from the CDN providers.

**Assumptions of Channel-based Security** As stated, the channel-based security model requires synchronous communication to maintain the secured channel. Nowadays, with the wide adoption of CDN and other types of services like load balancing service, DDoS mitigation service, and VPN, one single channel between the client and server no longer exists. Therefore, to ensure security with multiple channels, channel-based security models require that

- Each channel and middle node must be secure. If any channel is compromised, the security of the data exchange no longer holds.
- Additional trust relationship between client and middle node, or between middle node and server, must be established. For example, content providers share private keys for HTTPS connection to CDN nodes so that clients can trust the CDN nodes. Similarly, when using a VPN, the client must put trust in the VPN proxy as the VPN proxy can see all the incoming and outing traffic made by the client.

## 2.3 Data-centric Security

In contrast, another security model is the data-centric network security model as proposed by CCN/ICN/NDN (hereafter NDN because NDN is the most developed version so far). The unit being secured is a piece of named data, called a Data packet. A Data packet is composed of a data name, the data payload, and a digital signature covering the name and data payload. The signature is generated by the data producer application at the time of Data packet creation. The security properties of data, *e.g.*, the integrity and authenticity ensured by the signature, the confidentiality provided by payload encryption, are independent of any channels, Being named, a Data packet can be fetched by its name; the request carrying

the name is called an Interest packet. Being secured in a self-contained way, a Data packet can be cached by any network nodes, and even a Data packet is fetched from the cache rather than the original data producer, a data consumer application can still verify the data's security properties.

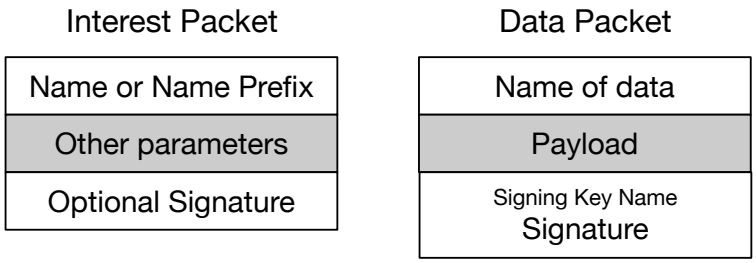



Figure 2.1: Named secured data in NDN

Besides being applied in NDN, the concept of this network security model can also be observed in a number of application layer projects such as IPFS, JSON web signature (JWS) [JBS15], Verifiable Credentials [SLC19], and Signed HTTP Exchange (SXG) [Yas20] used in the Web Package [Yas21]. For example, similar to a Data packet in NDN, a JSON with signature, a verifiable credential, or a signed HTTP file can be cached and consumed preserving the security properties without the need to establishing a channel to the original server where they originate. However, due to the currently deployed Internet architecture, they are not explicitly named for data retrieval; instead, to obtain such a secured data object, a client needs to establish channels to the data object host and use application layer identifier, *e.g.*, URL, to get the object. In comparison, IPFS has explicitly named secured data: each data object is named with hash value based content identifiers which are to identify content by what is in it rather than where it is located. Each data object in IPFS is also secured in a way that once the client obtains the content identifier, it can ensure the integrity of the fetched data object by comparing the hash value. Compared with NDN, the IPFS's hash value name only ensures integrity while a digital signature in an NDN Data packet also ensures authenticity. Importantly, since IPFS is built over the current TCP/IP architecture, it requires a mapping service to translate IPFS content identifiers to IP addresses.

Based on these characteristics, we can summarize the data-centric security model as follows.

- The security properties of data are bound to data itself.
- Security properties are preserved regardless of the network context.
- Data-centric security model allows both synchronous and asynchronous data exchange.

With the data-centric security model, each data object is secured independently of where the data is kept. Correspondingly, to make these data objects available to the network and preserve the independence of data security, the network identifier should also be data-centric, *e.g.*, the IPFS hash based name and NDN's data name. In the rest of the paper, we focus on NDN's implementation of the data-centric security model, namely, the named secured data.

**Data-centric Security Preserves End-to-end Principle** As stated, the use of channel-based security breaks the end-to-end principle since many direct end-to-end connections no longer exist. In contrast, NDN and generally speaking, the data-centric security model, still preserve the end-to-end principle Using NDN's Data packets as an example, no matter how a Data packet is obtained, the consumer application can verify its security properties that were directly added by the data producer application. Such security check is between two ends and unrelated to any other middle nodes like CDN services.

**Data-centric Security with Forward Secrecy** Forward secrecy with its typical implementation, TLS, gives an impression that forward secrecy is designed for channel-based security. This is because in TLS, the security is bound to the connection, and once the TCP connection changes, a new TLS session or a new set of keying material derived from the previously shared key (PSK) should be used.

However, does this mean the forward secrecy can only exist in channel-based security? To answer the question, we first revisit the definition of the forward secrecy [Wik21a].

> "In cryptography, forward secrecy (FS), also known as perfect forward secrecy (PFS), is a feature of specific key agreement protocols that gives assurances that session keys will not be compromised even if long-term secrets used in the session key exchange are compromised."

The core of FS is the relationship between a dedicated key for a session versus a long-term private key kept by the end hosts. This does not restrict whether the dedicated key is bound to a channel identified by a pair of sockets or a group of Interest and Data packets that are independent of channels (there is no concept of a channel in NDN).

Considering a producer and a consumer want to exchange some secret data that is not supposed to be seen by third parties. To ensure forward secrecy, they can perform a Diffie-Hellman (DH) key exchange to obtain a shared secret. Then both of them derive a common key for content encryption, *e.g.*, by applying AES-GCM to the payload of Data packets. On the one hand, the Data packet carrying the encrypted payload still promise forward secrecy because even if the long-term identity key of either the producer or the receiver gets compromised, the attacker cannot recover the shared secret to decrypt the data packet. On the other hand, the security is still data-centric because the security of the Data packet is independent of any underlying channels. For example, the data can be cached or stored by some storage services and be used asynchronously by the consumer.

The use of DH or other means to achieve forward secrecy does cause some limitations to the Data packets. For example, the data cannot be reused by multiple consumers and there will be a very low hit rate even the packet is cached. However, in the scenario where forward secrecy is needed, the data being protected is usually not expected to be reused, so the limitation can actually be beneficial to the security of the data.

## 2.4 A Comparison

From the description of the two different security models, the core difference is the network dependency of the security primitives that have been applied to the data. Under the channel-based security model, data security relies on the channel between the two hosts while in data-centric security model, the data security properties of data hold regardless of the network context. For example, a cached TLS packet with MAC and encrypted payload cannot be reused in another TLS channel but a cached NDN Data packet can be reused by other requesters.

The root cause of the difference comes from the different network abstractions used by the two architectures. TCP/IP is about the connection between a pair of hosts, so the security model added to it naturally bound to the channel. NDN centers around named data and thus the security model fits it by directly securing named data instead of channels.

As stated, there is a mismatch between the application's needs for secured data and the secured channels provided by the channel-based security model. When data objects need to be cached in CDN nodes near the content requesters, directly applying channel-based security becomes insufficient because the CDN providers' TLS certificates do not match the content provider's certificate. As a consequence, quick-and-dirty solutions like private key sharing between content providers and CDN providers are being used. In contrast, if the data-centric security can be applied, there is no mismatch and the system complexity can be much reduced because requesters can check the security of the content regardless of whether it is fetched from a CDN provider or the content provider itself. In fact, the Web Package mechanism [Yas21] was designed to solve this issue with an application layer mechanism, following the idea of data-centric security.

Through the comparison, we can see data-centric security model matches the need of content delivery applications. If being deployed in the current Internet, the mismatch between content delivery and channel-based security model can be well addressed. Nowadays,

more than half of all web traffic has been served over CDNs [Aka21], and that percentage is predicted to grow as applications offer more varied content types. Therefore, if not well addressed, the mismatch may keep causing new complexities and security issues in the future.

It is noteworthy that the comparison does not indicate data-centric security model is always more suitable than channel-based security. A typical use case of channel-based security is the exchange of data that is customized, non-static, and not reusable, *e.g.*, user's bank account information. For such type of data, cache or re-usability is not needed and usually unwanted, and thus a secured channel directly between the client and the server can be a better choice. Furthermore, in the channel-based security model, expensive public key signature verification is usually used only at the beginning of the connection for identity verification, and the later data is exchanged using lightweight message authentication code (MAC) and symmetric key encryption. In data-centric security, however, public key signature scheme is widely used and thus may affect the performance of data exchange.

## 2.5 Trust as a Central Piece of Network Security

Trust in security can be considered as an agreed-upon relationship between two or more identities that is governed by criteria for secure interaction, behavior, and outcomes relative to the protection of assets In the channel-based security model, when setting up a secured channel, the client must strictly authenticate the server to ensure the server is trustworthy with regard to the desired data. In the data-centric security model, the data consumer performs data signature verification to authenticate the data producer with regard to the corresponding named data.

Trust is a fundamental component in network security since network security relies on a certain trust relationship between involved parties. First, trust is directly reflected by identity authentication because trust only applies to certain identities. Without knowing the identity of the other side of a channel or the data producer, there is no trust relationship.

For example, a user Alice wants to retrieve news from ABC News. This already indicates that the user trusts the identity of ABC news for providing her the desired news data. If Alice cannot authenticate that the data is sent from (under channel-based security) or produced by (under data-centric security) ABC news, then Alice cannot verify the obtained data against certain trust policies. Second, other security properties like authorization are also built on certain trust relationships. For example, when performing SSH to a remote server, after authenticating the identity of the client, based on the trust relationship encoded in the client's user group configuration, the client will be authorized accordingly.

Trust relationship in cyberspace is built based on the trust relationship in the real world. For this reason, out-of-band operations are usually, if not always, needed to bring the trust from the real world to cyberspace. One most common way is to install keys associated with the trusted party, *e.g.* public key certificate, in an out-of-band way. Using the example of Alice and ABC News, the CA certificates used to authenticate ABC News' certificate are pre-installed to Alice's web browser.

**Trust and Identity Management in Today's Internet** In today's public Internet, an identity is represented by its TLS certificate. Until recently, most TLS certificates have been issued by a relatively small number of commercial Certificate Authorities (CAs) [ABC19]. Commercial CA services came into existence in the mid-90s to meet the need of cryptographic protection for the emerging e-commerce applications at the time. Pragmatic solutions were also developed to install the self-signed certificates of these CAs into end-user devices as *trust anchors*, largely through side channels (*e.g.*, web browser installation and updates), and behind the back of users. Rapid growth for certificate services also leads to the rise of hierarchical relationships between CA providers, for example with CA-1 issues a certificate to CA-2 which in turn issues certificates to end users. By and large one could view CAs in the web PKI as trust anchors, whose certificates get installed into user devices. Note that in practice, a trust anchor can run multiple intermediate CAs whose certificates are derived

from the trust anchor certificate, but they usually managed by the same party and thus can be considered as one entity.

The research community has identified several longstanding issues with today's CA practice, which we summarize below.

- **Authenticity vs. Trust**: There are different views regarding the meaning of a certificate $C_P$ issued to party $P$ by a CA. Most commercial CA providers hold the view that if $P$ holds a certificate $C_P$ issued by them (especially if it is an Extended Validation/EV certificate), then $C_P$ certifies *both* $P$'s identity and its trustworthiness. Another view, shared by LetsEncrypt, today's de facto largest CA issuing free-of-charge certificates using automated means [Let20, BHM19], believes that a valid TLS certificate only certifies the authenticity of party $P$'s identity, but not $P$'s trustworthiness. For example, a phishing website can successfully obtain a valid TLS certificate from a CA, commercial, or otherwise, as long as it owns a domain name and has an available web server [Cim17].

- **Externality and Trust**: Today's CAs are *external* and thus agnostic to the communicating parties who need to establish trustworthy relations [SJ09]. Anyone can get a certificate, $n$ unrelated parties $P_1$, $P_2 \cdots P_n$ possessing a certificate each is unrelated to whether they trust each other. The fact that the trust anchors of these certificates are decided by yet another *external* entity (*e.g.*, browser vendors) behind the back of end users further argues that certificates serve the purpose of authentication at best, not trust relations.

- **Constraints**: In today's practice, a certificate is issued to a site (a company, or at least a DNS name owner), and the certificate owner is not allowed to issue certificates for its own sub-namespaces, making it difficult to support the principle of "minimal privilege". Another practice is setting a prolonged certificate lifetime: generally a few months as a minimum, and often one year or even longer–this is especially true in the case of

manually processed certificates. Compounded with the coarse certificate granularity, a long lifetime reduces certificates' resiliency against brute force usage-analysis attacks.

**Distributed Trust Management and Flexible Trust Policy** In comparison to the current practice, another way of managing trust and identity is to let each system have its own trust anchor and identity management, for example, with Simple Distributed Security Infrastructure (SDSI) [RL96]. Under this model, the trust management is internal of the system and as a consequence, the identity manager can issue certificates to entities based on their application-layer attributes. For example, each certificate can be associated with a meaningful name that can be understood by other entities in the same system. By embedding different attributes into the name, the certificate can allow fine-grained security. For example, if the name carries the role of the entity, the certificate can be used for role-based access control.

To overcome the inability to issue sub-namespace certificates while not messing up the system with a large number of certificates, expressive trust policies should be made for each critical operation. These trust policies should clearly constrain the power of each identity and their derived identities with regard to certain application layer operations. Regarding the lifetime of certificates, to allow short-lived certificates as an enhancement of security, automated certificate management is required.

# CHAPTER 3

# Data-centric Security Support in NDN

## 3.1 A Brief Introduction to NDN

Named Data Networking (NDN) [ZAB14] changes the Internet communication model from TCP/IP's pushing packets to destination addresses to fetching named secured data. Figure 3.1 shows the most important features of NDN.

- Naming data with application-defined names. Data names can provide rich application layer semantics.
- Securing data directly. This allows security stays with the data regardless of the network context.
- Stateful forwarding to forward data requests and the replied data in the network.

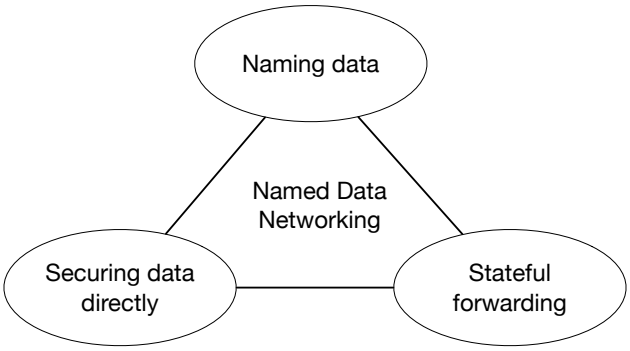



Figure 3.1: Three most important features of NDN

To be more specific, in NDN, an application fetches a piece of data by requesting its name. Such a request is called an *Interest packet* and the fetched data is a *Data packet*.

As shown in Figure 3.2, both Interest and Data do not contain any IP addresses or other network topological identifiers.

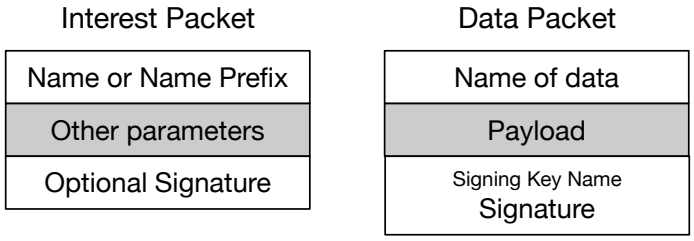
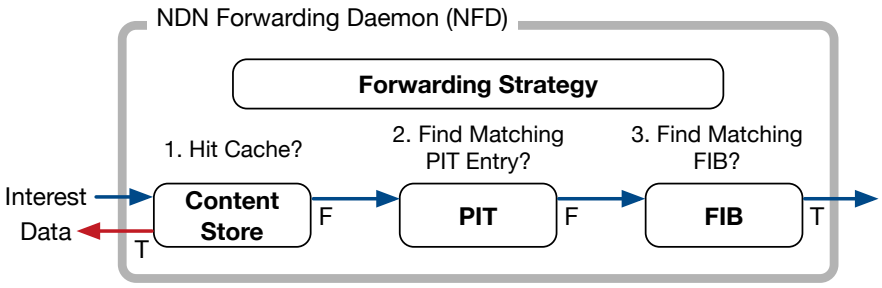


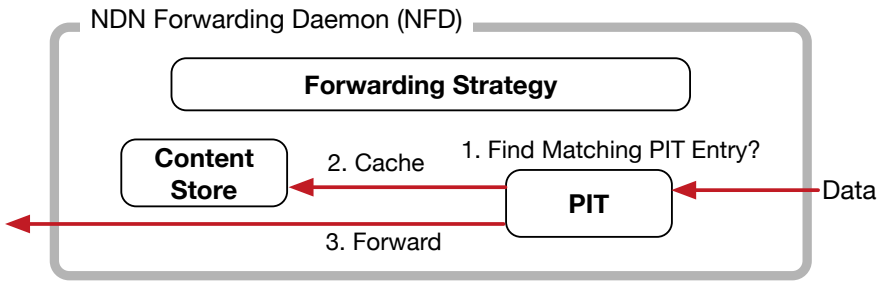

Figure 3.2: Interest and Data packets in NDN

After receiving an Interest packet from the network, as shown in Figure 3.3, a router forwards the Interest by looking up the Interest name and performing long-prefix match against the forwarding information base (FIB), which is like the FIB in IP but is made up with name prefixes instead of IP prefixes. For each forwarded Interest packet, the router will set up a record of the Interest and its forwarding path and keep it in a data structured called Pending Interest Table (PIT). After the Interest packet arrives at a network node or end host that has the matched Data packet, the Data packet will be replied. To forward the Data packet back to the Interest sender, each router will look up the PIT, find the corresponding Interest record, and send the Data packet to the interface from which the Interest came. In this way, the Data packet will following the path of Interest reversely back to the application. At the same time, routers can cache the Data packets so as to satisfy future Interest packets asking for the same piece of data.

Therefore, data name is the network identifier used in NDN. However, different from TCP/IP's design, where each layer defines its own identifier space (*e.g.*, network layer uses IP while transport layer uses port number), data names in NDN are semantic meaningful, hierarchical, and most importantly, defined by applications.




(a) Forwarding an Interest packet




(b) Forwarding a Data packet

Figure 3.3: Network packet forwarding in NDN

## 3.2 Security Support of NDN

NDN as a network layer protocol itself does not directly provide data security. Instead, NDN builds security into its design by allowing applications to add security to their data in a data-centric way. To be more specific, at the time of Data packet generation, the data producer will need to cryptographically sign each Data packet. In addition, cryptographic schemes like symmetric or asymmetric key encryption or system solutions like onion routing can be applied over NDN to provide data confidentiality with access control and privacy.

The security support in NDN security is realized in a cross-layer manner: NDN provides a general format to carry the security properties with Data packets while applications make use of it by adding/verifying security with keying material and security policies. Since data names provide rich application layer context, names can be utilized for security policy making.

### 3.2.1 Data Availability

NDN provides availability naturally by using named secured data as the basic unit of networking. Specifically, since each Data packet is secured directly, a Data packet can be cached by any network router or dedicated storage services, and at the same time, does not invalidate the data security, thus improving the availability of data. For example, considering Data packets that carry static content, even if the original producer is offline, data can still be fetched by its name if the Data packets have been cached in the network.

### 3.2.2 Authentication and Data Authenticity

In NDN, a Data packet is not only named but also secured and the authentication of data is built over the public key cryptography. To be more specific, at the time of generation, the producer application will cryptographically sign the packet. This allows any party who knows the public key of the producer to verify the integrity and authenticity of the packet. Importantly, this verification is independent of where the packet is fetched from (*i.e.*, from a router's cache or from the producer application).

To realize the above ability, NDN producers require at least a name for generating data under and a public/private key pair. For this purpose, NDN assumes the distributed name management and SDSI [RL96], where the Internet is composed of a number of organizations where each organization manages its own namespace and maintains its own public key infrastructure (PKI). This is in a similar way to today's domain name system in which each authoritative server can manage their own DNS names in the zone.

Under such a model, each NDN entity is supposed to obtain a name, called an identity name, from the local name manager. Importantly, the identity name will be associated with the entity's public key. To be more specific, the entity will generate or reuse a pair of public and private key, and apply for a public key certificate from the local certificate manager. An NDN certificate, which is also a regular NDN Data packet, binds the identity name and the

public key together, and is signed by the certificate manager. As a Data packet, a certificate is named in the following way and can be fetched via Interest packets.

/<identity name>/KEY/<key ID>/<Issuer Info>/<Certificate ID>

The difference between an NDN certificate manager and a commercial TLS certificate authority (CA) is as follows.

- Since an NDN certificate certifies a name and the name is semantically meaningful. An NDN certificate can not only be used for authentication but also can indicate authorization. For example, the owner of the certificate whose identity name is /example.org/manager indicates a higher level of access rights than the owner of the certificate for /example.org/guest. In contrast, a TLS certificate usually only serves the purpose of authentication. This is because a commercial TLS CA is not in any specific application system and lacks the semantics that is required for authorization.
- An NDN certificate manager is managed by the local organization while the TLS CAs assume a global trust model because the CA's trust anchor certificates are globally installed by the Internet users.

An implementation of NDN certificate management is NDNCERT [ZAZ17, ZS20].

It is noteworthy that, though NDN provides approaches to data authentication, the security of data is more than that. Authentication and trust verification ensures the data is truly signed by the claimed private key and the key is authorized to sign such a packet. However, it does not guarantee the correctness of the content; for example, the private key of the data producer can be compromised and is used to sign incorrect content.

### 3.2.3 Named based Authorization

Since both the Data packet and the data producer have a name, constraints on these names can be established to express trust policies for authorization. In NDN, these constraints are

called trust schema [YAC15]. To be more specific, a trust schema rule contains (i) the packet name that the rule applies to, (ii) the expected signing key name or name pattern, and (iii) a number of NDN certificates as the trust anchors. If a packet's signing key does not satisfy the rule, or if the signer's certificate cannot be verified against an allowed certificate chain (*e.g.*, each certificate in the chain is allowed by the trust schema), the packet will be rejected even if the signature value itself is valid.

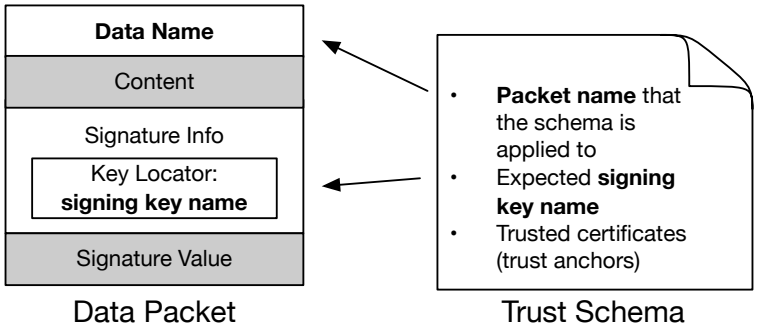



Figure 3.4: Trust schema defines the expected data and key names

In this way, besides the signature verification (*i.e.*, authentication), the trust schema further restricts which key can be used to sign which type of data (*i.e.*, authorization). The trust schema rules are supposed to be made following the least privilege principle [SS75]. For example, a temperature sensor is only allowed to sign the temperature data but is not authorized to generate and sign a command to unlock a door. An example of using trust schema in smart homes is Sovereign as introduced in Section 4.1.

As a generic component, the trust schema is also used in other security supports. For example, in the data encryption based access control, the trust schema can be used by the access control manager to define which consumer can request which decryption key.

### 3.2.4 Data Confidentiality and Access Control

In NDN, confidentiality can be implemented by encryption with access control based on decryption key issuance. Importantly, as stated, public keys carried by certificates are named

in NDN and can be fetched as normal NDN Data packets. This is the same for encryption and decryption keys. The difference is that the decryption key is further encrypted so that only authorized consumers can access the decryption key. In addition, hybrid encryption is usually used. That is, the payload is encrypted with a symmetric key for efficiency and the symmetric key is encrypted with the asymmetric key which separates the role of encryptor and decryptor (in comparison, the symmetric key does not distinguish encryptor and decryptor).

In this way, the access control can be reduced to the management of decryption key issuance. The underlying realization varies depending on whether the decryption key issuance is synchronous or not. Here we give two possible approaches as an example.

- If such issuance is synchronous, the decryption key can be requested through a remote procedure call by the consumers. In this process, the access manager can check the trust schema to decide whether such a consumer can sign such a decryption key request.
- If the application needs the key issuance process to be asynchronous for better availability, the key manager can generate an encrypted decryption key for all consumers in advance and name these keys following pre-defined naming conventions. Therefore, each consumer can follow naming conventions to fetch the key that belongs to the consumer back without contacting the key manager in real time.

An implementation of the asynchronous NDN based access control is Name-based Access Control (NAC) [ZYR18] which supports RSA encryption and attribute-based encryption [BSW07] of NDN Data packets with automatic key delivery with names.

### 3.2.5 Content and Name Privacy

Privacy poses a higher requirement than applying encryption to the Data content. As defined in ISO 27701 [ISO19], privacy refers to the term that describes the end result of adequate controls over the 'processing' of Personal Identifiable Information (PII) [MGS10a]. In the

context of attribute-based authentication, privacy is defined through unlinkability, which refers to the need to be able to handle and process personal information anonymously, in a way that precludes being able to identify the original data subjects from the information being communicated and processed. Therefore, only encrypting content does not preserve unlinkability if the name of the packet can be linked or traced.

A further privacy enhancement mechanism is name obfuscation, which aims to minimize the information disclosure from names. From the use of NDN names, three types of information are revealed.

- Name or prefix that can indicate a host.
- Name or prefix that can indicate an application within the host.
- Name that can indicate a piece of content served by the application.

This is similar to the three identifiers used in TCP/IP, namely, IP address, port number, and application-layer identifier. Among the three types of information, the content name is most diverse and information-intensive. In addition, an NDN name prefix used to indicate host or application can also bring more information compared with an IP address or a port number. For example, /west-la-medical-center/emergency-service can reveal more information than "1.2.3.4:80". Even though the IP address can be mapped to the domain name "west-la-medical-center" after a reverse DNS lookup, the /emergency-service is still more meaningful than "80".

Common approaches to name privacy can include (i) encryption based mechanisms where the sensitive information in name will be encrypted, (ii) proxy based mechanisms where the proxy will send the requests like how VPN and Onion Routing work in TCP/IP.

# CHAPTER 4

# Usable Security Solutions with Named Secured Data

In this section, we introduce several network systems where their security challenges can be addressed in a more efficient way by NDN's data-centric security model than the conventional channel-based security. Across very different network environments (*e.g.* home network, vehicular network, Internet) and different application purposes (*e.g.* access control, content delivery, DDoS mitigation), we show that NDN's named secured data provides a solid foundation to address some security challenges with the following properties.

- Name allows fine-grained security policy making and key management.
- Securing data directly allows asynchronous communication and is friendly to non-infrastructure networks like wireless IoT.
- Stateful forwarding, which is to work with named secured data, provides valuable state and insights of the traffic.

## 4.1 Sovereign: NDN Based Smart Home System

The majority of the materials contained in this section were modified or quoted verbatim from [ZGM20].

Sovereign is an NDN based user-controlled smart home system. Different from the current practice, namely cloud-based smart home, Sovereign removes the dependency on the external servers from the control loop and allows secured local communication. The insight

of Sovereign is to utilize NDN's named secured data to facilitate decentralized local wireless communication and end-to-end security.

### 4.1.1 Background and Motivation

Most of today's smart homes depend on an external cloud service to operate. A technical reason is the deployed network architecture, namely, the channel-based TCP/IP communication model. This communication model assumes a well-established and synchronous connection between two hosts. However, in a smart home network or IoT in general, maintaining mesh connection among participants is not easy, *e.g.*, it is required to maintain the mapping between resources/services to sockets. As a result, a less-resistant way is to utilize a centralized node for home participants to connect to and communicate with others. Besides the technical reasons, the existing deployment of the cloud infrastructure also allows cloud service providers to serve smart homes without changing much of the existing hardware or software stack.

With such a model, devices are managed by the cloud backend and end users utilize the interface exposed by the cloud to use the home smart appliances. However, recent reports [TST17, LB16, UJS13, CCJ11, SWA17] indicate a number of issues in the cloud-based smart home systems. For example, the cloud backend can know all the commands made by the home users, which reveals a huge amount of user privacy at home.

Our goal is to preserve home users' privacy from the cloud and other external dependencies with a role switch of home users and the external parties. To be more specific, we propose Sovereign to provide secure device-to-device (D2D) communication which is directly guided by the user-defined security policies instead of the cloud. For this purpose, we let all home participants trust a local trust anchor certificate that is locally kept by the user's device. In addition, we also need a local entity, called a local controller, as the interface for users to configure security policies. To fully utilize the broadcast media, the D2D communication is based on NDN's named secured data so that security is no longer bound to one-to-one

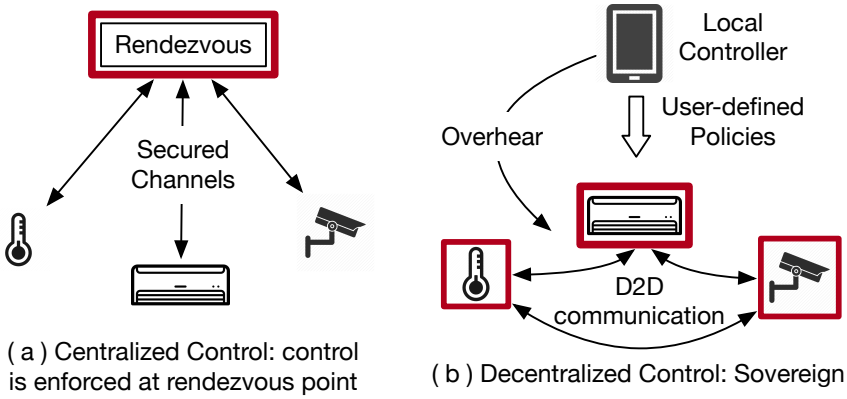



Figure 4.1: Two different models

channels. To enforce security in distributed D2D communication, we make individual home participants smarter by installing necessary security policies and keying material from the local controller. Putting all pieces together, home participants are able to communicate directly and securely under users' control, removing the single point of failure resulted from the centralized message broker.

This design is enabled by the current technological advance that the micro-controller market now turns to 32 bits, which provides sufficient memory and computation for basic cryptographic operations needed by Sovereign [Mar21, IC 15, IC 18a, IC 18b, EE 13].

Regarding the use of the cloud, Sovereign will not prevent utilizing cloud resources for computation-intensive tasks or reliable data storage. However, the difference is that Sovereign treats the cloud or other external services as regular devices or applications managed by the local system. That is, these external services must be explicitly allowed and authorized by the users, obtain the corresponding identity and keying material, and then access the home data or services.

### 4.1.2 Design Overview

In contrast to the current practice where the trust anchor and the security management authority are in the cloud, Sovereign changes it in a fundamental way by bringing the trust

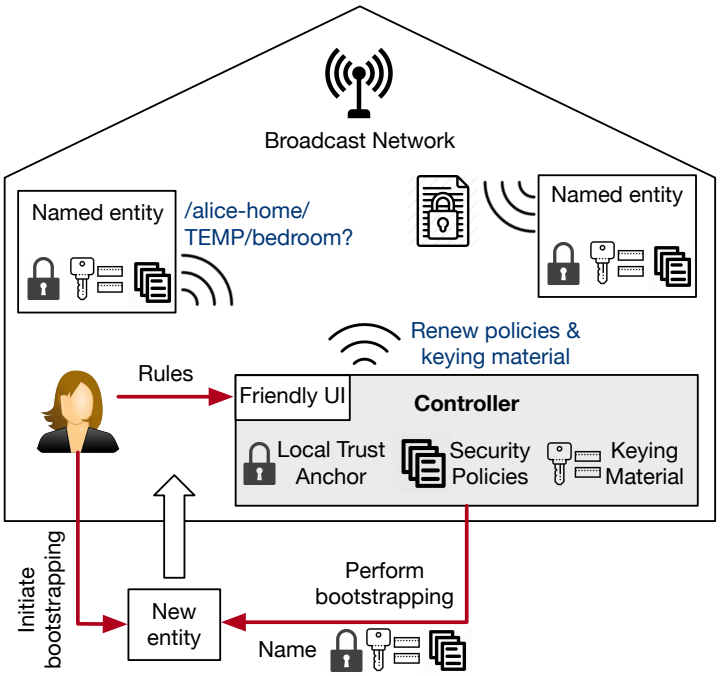



Figure 4.2: An overview of Sovereign

anchor to the Sovereign home controller, which is located at home and directly managed by the home users. Home participants, including devices and applications, directly follow the security policies made by home users through the local controller and communicate with others. The communication is directly over the wireless broadcast media in NDN.

Figure 4.2 shows the overview of our design. Consider a home user who purchased a new air conditioner (AC) for her home and wants to add it to her Sovereign system. To start with, the user initiates the device bootstrapping process by some out-of-band operations like QR code scanning or pressing some buttons. In the bootstrapping, after verifying each other, the device will register itself to the controller and the controller will (i) send the system trust anchor to the device, (ii) sends keying material and security policies needed by the device, and (iii) assigns a name to the device according to the naming conventions. In Sovereign, security policies are based on trust schema rules that are converted from user-decided configurations collected by the controller.

After that, the AC will start running by tracking and adjusting the home temperature,

| Type | Naming Convention |
|---|---|
| Device&Application | /<home-prefix>/<service>/<location>/<entity-id>*<br>*e.g.*/home/AirCon/bedroom/north-ac-1 |
| Commands to executables | /<home-prefix>/<service>/<**scope**>/CMD/<cmd-id>**<br>*e.g.*/home/AirCon/**bedroom/north-ac-1**/CMD/set-temp – device-level command<br>*e.g.*/home/AirCon/**bedroom**/CMD/set-temp – room-level command<br>*e.g.*/home/AirCon/CMD/set-temp – home-level command |
| Service's Content | /<home-prefix>/<service>/CONTENT/<location>/<entity-id>/<content-id>**<br>*e.g.*/home/TEMP/CONTENT/bedroom/senor-1/temp |
| Encryption/Decryption Key | /<home-prefix>/<scope>/EKEY<br>/<home-prefix>/<scope>/DKEY |
| Security Policy | /<home-prefix>/RULE/<location>/<entity-id>** |

Notation: A component with <> represent a variable. A component without <> represent a constant string component.

∗: Actual service command, content, and policy NDN Data packets will have a timestamp suffix to achieve the data uniqueness.

∗∗: The corresponding identity key name is the identity name with a "KEY/<key-id>" suffix.

Table 4.1: Naming Conventions in Sovereign

where home temperature data is generated by home temperature sensors. In addition, we assume the AC can also close bedroom windows. In Sovereign, the temperature data fetching and window command sending all take place directly between the involved devices over the broadcast network in the form of NDN Interest–Data exchange. To get temperature data, the AC directly broadcasts the desired data name so that any other entities who have the data can reply. After fetching the Data packet carrying the temperature, the AC will first verify the packet is signed by another home entity, whose certificate is endorsed by the home trust anchor. Then, to access the payload, the AC needs to first decrypt it using keys obtained from the controller. When issuing a command to the window, the AC will also name this command according to the naming convention, encrypt the payload, and sign it with its own private key. Therefore, windows in the system can fetch the command by the name. If the AC is allowed by security policies to generate such a command, home windows will successfully verify, decrypt, and execute the command

### 4.1.3 Naming Conventions and Name-based Security Policies

In Sovereign, each piece of data, executable, device, and keying material is named following pre-defined naming conventions as shown in Table 4.1. The naming conventions are pre-installed with the system software and thus, each application knows how to construct an Interest packet to fetch the desired data or to invoke services.

Using the example of the bedroom air conditioner, the air conditioner listens to requests under three prefixes as follows so that the AC can react to commands of different levels, namely, device-level, room-level, and house-level commands. As shown, authorized entities can control the home temperature in the desired granularity.

/home/AirCon/**bedroom/north-ac-1**/CMD/set-temp
/home/AirCon/**bedroom**/CMD/set-temp
/home/AirCon/CMD/set-temp

Naming conventions also facilitate the expression of security policies. Security policies specify entities that are authorized to perform certain operations. In Sovereign, flexible control can be achieved by specifying the names of entities and resources in the policies. Given the name $P$ representing one or multiple entities, and the name $R$ representing one or multiple resources. One can limit $P$'s authorized actions to $R$ by defining a security policy. Such a policy can be written as a triple based on $P$ and $R$, which can be either be specific names, name prefixes, or regular expressions of names.

<$P$'s name, verb, $R$'s name>.

For example, the verbalized policy "the controller can command all door locks" can be written as the name-based policy <controller name, produce, door lock command prefix>. As another example, the verbalized policy "temperature sensors can produce temperature data" can be represented by the policy <prefix of temperature sensors, produce, temperature content prefix>.

### 4.1.4 Distributed End-to-end Security Enforcement

The authentication and access control are managed by the controller but enforced distributedly in D2D communication. In general, after converting user-defined rules into name-based security policies, the controller distributes these policies among all entities. During runtime, all Data packets are signed and encrypted by producers, and verified and decrypted by consumers. In the verification step, consumers consult the available security policies and verify whether Data producers are authorized for signing before consuming the content or executing a command.

**System Setup Phase**: In the system setup phase, the controller generates an asymmetric key pair. The public key is bound to the home prefix and published as a self-signed certificate. This certificate represents the home's trust anchor. The trust anchor will be installed on each entity during the entity bootstrapping. The private key is kept secret by the controller and is used to sign security policies, cryptographic keying material, and certificates for new entities.

**Entity Bootstrapping Phase**: In the entity bootstrapping, a new entity joins the system and learns cryptographic keying material and security policies required for later communication. The process takes place in the default broadcast media and starts with mutual authentication between the controller and the entity, in which out-of-band operations may be needed (*e.g.* QR code scanning). Then, the following information is secretly exchanged:

- The trust anchor certificate is installed on the new entity.
- The controller assigns the new entity an appropriate name according to the naming convention (supported by additional input from the homeowner, *e.g.* entity location).
- The controller issues a public key certificate binding the entity's public key and name together. The certificate is signed with the private key of the trust anchor.

- Security policies and cryptographic keys used for access control are transferred to the new entity.

An implementation of Sovereign's entity bootstrapping process is discussed in [Ano19].

**Enforcing Security Policies**: While name-based security policies are maintained by the controller, individual participants enforce these security policies for the D2D communication. To elaborate on how these policies are enforced, we differentiate between two types of security policies: (i) *produce*-policies define which entities are allowed to produce data of a specific name pattern (*e.g.* sending an actuating command to a door lock), (ii) *decrypt*-policies define the entities that are allowed to access data of a specific name pattern (*e.g.* data from specific sensors).

*Produce*-policies are enforced by data receivers. To be more specific, after authenticating a Data packet, the receiving entity extracts the data name and producer's name from NDN Data's name and signature fields. These names allow checking whether the data was generated by an authorized entity defined in security policies. For example, the following policy defines that temperature content can only be signed by a temperature sensor, where /alice-home/TEMP is the shared prefix of all home temperature sensors, and /alice-home/TEMP/CONTENT is the common prefix of temperature content.

</alice-home/TEMP, produce, /alice-home/TEMP/CONTENT>

Combining the Data's name, the producer's identity, and the available security policies allow receiving entities to reject temperature content produced by unauthorized parties.

The same procedure is applied for restricting entities to issue commands. For example, a *produce*-policy as follows indicates that all the automation applications running on the home hub named "hub-1" can invoke executables whose names match the specified regular expression[1].

[1]The regular expression <>* matches zero or more name components.

</alice-home/AUTO/hub-1, produce, /alice-home/LOCK/<>*/CMD>

Before executing the issued command, the receiving entities first check the verified producer name against the available security policies and rejects the command when issued by unauthorized entities.

*Decrypt*-policies are enforced by utilizing Data encryption. To be more specific, every entity is able to fetch Data by emitting an Interest packet carrying the desired content name. The encrypted data, however, can only be accessed when having access to the correct decryption key. In Sovereign, the controller is maintaining all encryption and decryption keys and allows authorized entities to obtain required keys. This reduces access control to maintaining the access of corresponding decryption keys. For example, the *decrypt*-policy "bedroom AC can read the temperature" is written as follows:

</alice-home/AirCon/bedroom, decrypt, /alice-home/TEMP/DKEY>

The subject name /alice-home/TEMP/DKEY represents the decryption key to the content produced under the temperature service.

Encryption and decryption keys are distributed during the entity bootstrapping process. However, keys can be renewed or changed (*e.g.* for access right revocation purpose), and hence, entities need to securely retrieve the updated keys from the controller. In Sovereign, decryption keys are encrypted for every authorized entity using the shared secret between an entity and the controller. Those secured decryption keys can then be retrieved using Interest-Data exchange. Renewed keys can also be kept in a storage component to make the smart home system resilient against temporary controller failures.

### 4.1.5 Integrating Sovereign Framework with Pub/Sub

The core implementation idea of Sovereign is to use a publish-subscribe (pub/sub) communication module that encapsulates naming, security, and networking primitives in one API.

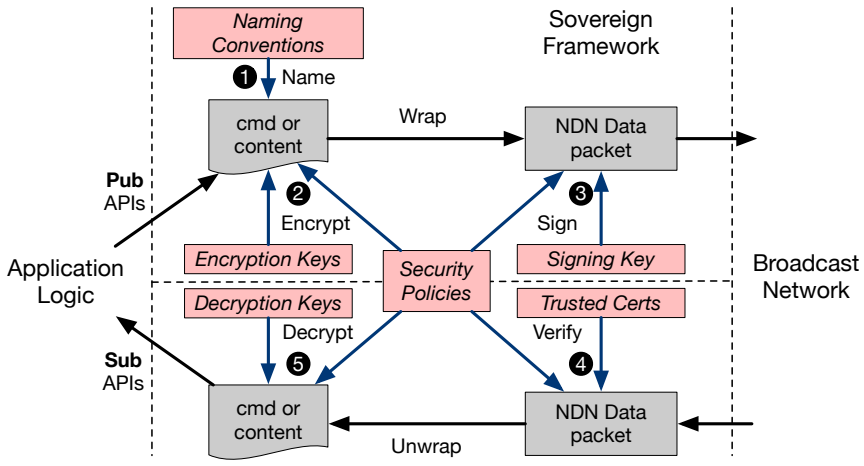



Figure 4.3: Workflow beneath the pub/sub API

Pub/sub is a messaging pattern that categorizes messages into semantically meaningful topics and is often seen in the context of IoT. Message producers (called publishers) publish messages to topics without knowing the set of message consumers (called subscribers). Subscribers choose to receive messages under pre-defined topics without the need to know the actual message producers. Pub/sub is used for two main reasons. First, pub/sub is data-centric which matches the data-centric networking and security design of Sovereign. Second, pub/sub has been widely adopted in existing IoT frameworks, and hence, adopting pub/sub provides convenience for developers.

Pub/sub API is directly built over Sovereign's D2D communication as presented in the previous section by handling name prefixes as topic identifiers. That is, subscribers use name prefixes to decide whether a message is under a certain topic or not. For example, a message carrying temperature data of the bedroom is mapped to the name prefix /home/TEMP/CONTENT/bedroom. This name prefix is further used as the pub/sub topic identifier. In this way, producers that publish content or commands under a topic is to generate Data packets named under the corresponding prefix. Subscribing to a topic is implemented by issuing Interests containing the topic's name prefix to fetch relevant data.

As indicated in Figure 4.3, in Sovereign implementation, the pub/sub API embeds nam-

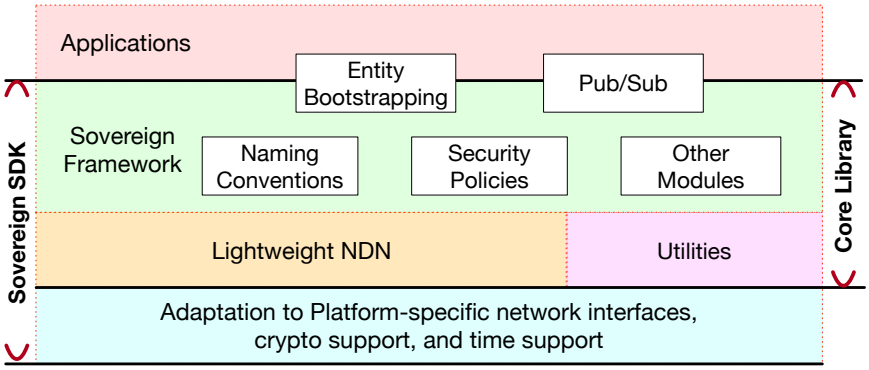



Figure 4.4: Structure of Sovereign's smart home SDK

ing, security, and networking considerations to make them transparent to developers. We illustrate the underlying workflow with an example, where an application issues a command to set the bedroom temperature to 70°F. After calling the publish API, Sovereign defines the command name based on application parameters following naming conventions (❶). Sovereign identifies an encryption key according to the topic and encrypts the payload (❷). Further, the API wraps the command name and the encrypted payload into a Data packet. Finally, the API uses the application's private identity key to sign the Data (❸) and makes it available on the local network. On the receiving end, after subscribing to a topic, the API automatically fetches the relevant Data packet. Once a Data packet is received, Sovereign verifies its signature and checks against security policies (❹). In the next step, the Data is decrypted with the corresponding decryption key (❺). Once verified and decrypted, the command is delivered to the application.

Sovereign is made available as an open-source, cross-platform software development kit (SDK). To allow use over constrained devices, the SDK is built on C and uses static memory allocation only. The SDK's structure is visualized in Figure 4.4. The core library implements the Sovereign framework. As indicated, the only components exposed to developers is the *Entity Bootstrapping* and the *Pub/Sub API*. Other components are transparent for developers. Moreover, the SDK includes an adaptation layer making the core library work across different platforms and communication media. So far, the adaptation layer is tested

for platforms including Linux/Unix, RIOT OS [BHG13], and Nordic NRF boards [Nor21]. Regarding connectivity, the adaptation layer allows using Bluetooth, IEEE 802.15.4, and the legacy TCP/UDP used as link layer protocols.

Also, the Sovereign SDK includes a standalone NDN stack that is lightweight enough for constrained devices. This implementation is required since the official NDN library and forwarder — ndn-cxx [Ndn21] and NFD [NFD21] — are not designed for being used on constrained IoT devices. We have verified its compatibility to the official NDN implementation with full examination.

To be used in real smart home products, device vendors and application developers need to program with Sovereign SDK and write the program together with all dependencies to the hardware platform (*i.e.* the program memory on the microcontroller).

### 4.1.6 Evaluation

We first discuss how Sovereign can work with security and privacy. Thereafter, we evaluate Sovereign's performance and it shows Sovereign's low overhead, low memory and flash requirements, and the ability to work with constrained devices.

**Privacy and Security Assessment: Case Study and Comparison** We assess Sovereign's privacy and security by analyzing packet flows in two real-world applications. We select two simple yet representative open-source applications [Sma20, Sma15] taken from the official SmartThings GitHub repositories [Sma21], and realize the same functionality in Sovereign. The first application is designed for a smart switch and the second is an automation applet turning on a switch when a contact sensor is touched. The code snippets of the original programs (code block 1 & 3) and their Sovereign-based equivalents (code block 2 & 4) are compared in Figure 4.5. The switch device application changes its own state to "on" when it gets a turn-on command, and the automation applets subscribe to the state of a contact sensor and turn on the switch when the contact sensor is touched.

In the SmartThings device application (code block 1), the first line of the code securely connects the device to the home's cloud-backend. After that, the cloud recognizes the new device, learns its profile, and registers it to the device database for the home on the cloud. In contrast, the first line of the Sovereign device application (code block 2) bootstraps the device to a local controller. All sensitive information that is transmitted in the bootstrapping phase stays in the local network and is protected by encryption.

The first line of the automation applet of the SmartThings (code block 3) and the Sovereign application (code block 4) subscribes to a given topic. However, the underlying operations differ: the SmartThings application notifies the cloud backend about its interest in the given service, while the Sovereign application starts listening to data published under the given name prefix in the home network. Similarly, when the SmartThings application turns on the switch, the command message is sent to the cloud backend, where the command is verified and sent back to the switch in the local home. In contrast, publishing a command in Sovereign means producing a new Data packet and making it accessible in the local home network.

Reflecting the above comparison, we see that Sovereign provides the same functionality as cloud-based smart homes via local and secure communication. The home activities and data are not accessible to the cloud unless users explicitly grant access rights to a cloud service.s

**Latency and Memory Benchmark** To evaluate the performance of Sovereign, we measure the latency and memory overhead of the operations *entity bootstrapping*, *content delivery*, and *command delivery*. We conduct our evaluations using a Intel Core i7 laptop, a Raspberry Pi 3B (RPI) and an nRF52840 board [Nor21] that mimic smart home entities with different capabilities. RPIs are widely used in IoT projects and often used for prototyping. Those boards are equipped with an ARM Cortex A53 @1.4GHz processor and 1GB RAM. We classify the evaluated nRF52840 chip with its 32-bit Cortex M4@64MHz CPU, 1MB

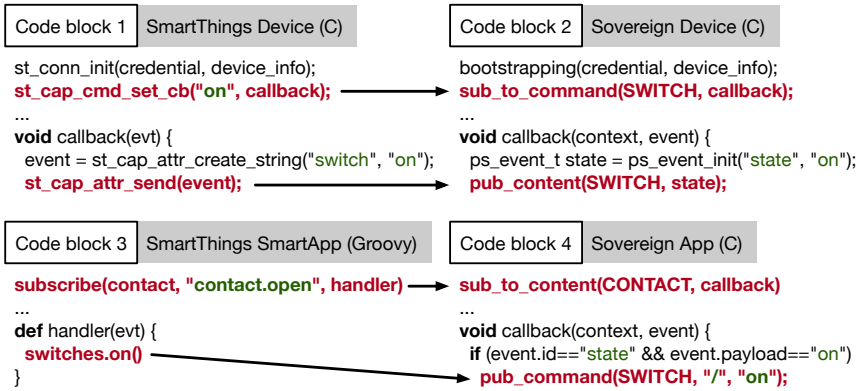



Figure 4.5: Code snippets in SmartThings and Sovereign

ROM, and 0.25MB RAM[2] as constrained hardware. The experiments with RPI is over WiFi connectivity and use a laptop equipped with a Core i7 processor as the Sovereign controller. Evaluation involving the nRF52840 is conducted over IEEE 802.15.4 and uses a simplified controller installed on another nRF52840.

Note that in the experiments of content and command delivery in Sovereign, we first used the controller to bootstrap a publisher and subscriber application, respectively, and then brought down the controller. Thus, the communication happens in a D2D manner without a controller involved, showing Sovereign's ability to continue operation even when the controller is temporarily down.

**Latency of Common Operations**: In the first experiment, we compare the latency of the above-described operations in the Sovereign and the AWS IoT framework. Therefore, we deploy a Sovereign Controller in the local wireless network. For the AWS IoT implementation, the location of the cloud-based AWS IoT controller is not under our control. However, to provide fair conditions, the wireless network is equipped with high-bandwidth Internet connectivity.

The leftmost bar-group in Figure 4.6 visualizes the latency for entity bootstrapping.

[2]We acknowledge that a part of current IoT devices shows lower capabilities. However, market studies [Mar21, IC 15, IC 18a, IC 18b, EE 13] have seen the market turning to 32-bit microcontrollers. Hence, we assume the nRF52840 as an appropriate candidate for future smart home systems.

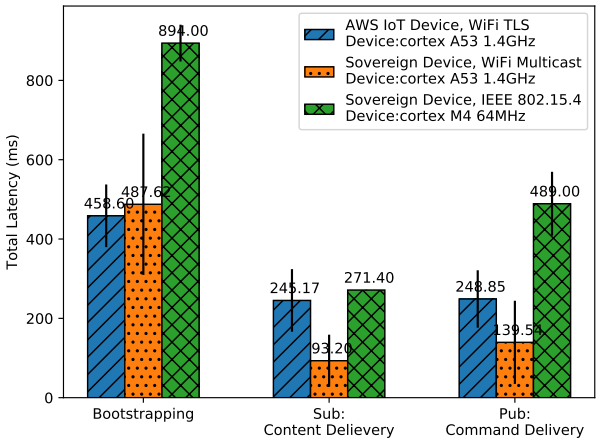



Figure 4.6: User-perceived latency of common operations in Sovereign and Amazon AWS IoT

Here, no significant difference between the RPI implementations of Sovereign and AWS IoT can be observed. The slower nRF52840 chip requires about twice as much time. Here, we want to highlight the use of heavy-weight cryptographic operations, such as key generation, that consume a significant amount of time on the constrained hardware.

The advantage of keeping communication local becomes clear when focusing on content and command delivery. Sovereign's RPI implementation is about 62% faster in content delivery and 42% faster in command delivery than the AWS IoT implementation on the same hardware. The constrained nRF52840 chip has about the same latency as AWS IoT for content delivery. For command delivery, the constrained device is slower than the RPI's AWS IoT implementation. However, a latency staying below 500 ms is assumed to be still practical for use in smart homes.

**Execution Time Breakdown**: A breakdown of Sovereign's runtime into individual operations is provided in Figure 4.7. The visualized operations are performed when preparing Data before broadcasting to the network and Data processing after receiving. This includes digital signature creation and verification (ECDSA), content encryption and decryption (AES CBC), security policy checking, NDN packet encoding/decoding, and other cryptographic

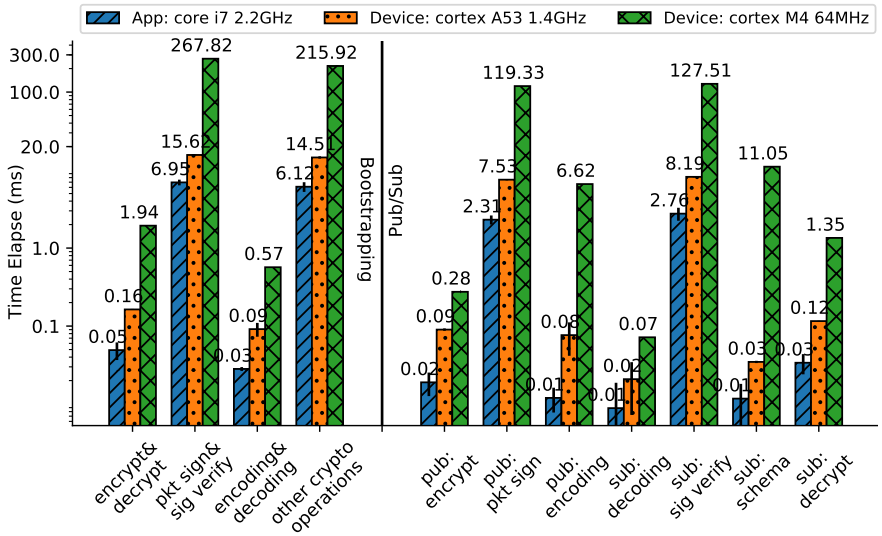



Figure 4.7: Breakdown of the execution time in Sovereign

operations (including ECDH, KDF). The results show that asymmetric cryptography consumes most of the computation time. This trend is observable across all evaluated devices.

**ROM and RAM Footprint**: We programmed an nRF52840 chip using RIOT OS [BHG13] for measuring Sovereign's memory footprint. Table 4.2 reports the size taken by individual Sovereign modules. All the main modules together require less than 50 KB of RAM and 70 KB of ROM. This indicates Sovereign's ability to be used on resource-constrained smart home devices.

## 4.2 FITT: DDoS Mitigation with NDN

The majority of the materials contained in this section were modified or quoted verbatim from [ZVO19].

Fine-grained Interest traffic throttling, or FITT, is an NDN based DDoS mitigation system. The core idea of FITT is to utilize NDN's stateful forwarding to traceback the attack resources and use names to achieve fine-grained traffic throttling. While the stateful forwarding itself is not part of the data-centric security model, it is designed to forward named

| Program/Modules | ROM Use | RAM Use |
|---|---|---|
| **Subscriber in total** | **62KB** | **47.3KB** |
| **Publisher in total** | **52.4KB** | **38.2KB** |
| Application | 1.8% | 7.3% |
| High-level Modules | 20.7% | 34.2% |
| Utilities | 3.3% | 14.4% |
| Crypto Tools | 25.1% | 0.2% |
| Network Forwarder | 24.1% | 25.0% |
| OS and Adaptation | 25.1% | 18.9% |

Table 4.2: ROM and RAM consumption of Sovereign applications

secured data in the network and thus closely coupled with NDN's data-centric security model.

### 4.2.1 Background and Motivation

DDoS attacks have bothered the Internet for decades and often utilize intrinsic properties of the TCP/IP architecture [OSZ20, prn18]. While the TCP/IP networking model has achieved great success, the other side of its design has also been abused by attackers to launch DDoS attacks. The ever-increasing size, frequency, and sophistication of DDoS attacks call for new approaches that can be partially or fully deployed imminently. We propose that this urgent need may accelerate our consideration of a new Internet architecture, and that evidence shows that there may now be economic *incentives* for large operators to upgrade their existing infrastructures to embrace it.

Indeed, starting with early DDoS attacks (*e.g.*, attacks from the Trin00 botnet in 1999 [CER99]) through to recent attacks from the Mirai botnet [AAB17], the remediation techniques used in today's Internet suggest that our defensive tactics may not be fundamentally keeping pace with attackers [OSZ20]. Rather, with attacking botnet nodes swelling in size to hundreds of thousands, and even millions, attacks have grown large enough that their attack volume rivals provisioned capacity of DDoS mitigation providers. The Mirai botnet serves as

a quintessential example, in that it was used to launch some of the largest DDoS attacks in history, and it did so using compromised devices that primarily included Internet of Things (IoT) devices and household appliances that were both easily discoverable and poorly protected [AAB17]. We note that DDoS has evolved to be *more distributed* than ever, and to increasingly using application-level semantics (e.g. reflective amplification attacks using DNS, NTP, memcached, etc.). On the other hand, service operators, providers, and mitigation services [Aka19, Neu19, Clo19] have had little recourse but to *centralize defenses* and backhaul or black-hole undisrupted attack traffic ("packet love") in large DDoS mitigation service networks. These DDoS mitigation approaches haul offending packets deeper into the network and require an ever increasing amount of deep packet inspection and state in terms of flow semantics to filter out attack packets, thus they do not scale well.

In this work, we examine the basic functions in NDN, that can address the principle weaknesses in today's IP networks and try to mitigate DDoS with properties provided by NDN. Importantly, to allow this to be used in real-world scenarios, the new approach should be incrementally deployable and align the incentives of existing service providers.

### 4.2.2 TCP/IP Architecture as the Root Cause of DDoS

Most DDoS attacks that are launched on today's Internet are made possible by utilizing features in the TCP/IP network architecture. Previous work [JWS03, BCR16, YWA05, HG04, Ros14] observed that DDoS attacks often exploit the following specific properties in IP:

- *Push-model Communication*: Any Internet node can send packets to any other IP address. This leaves DDoS attack victims with no way to stop the attack traffic.
- *Destination-based Delivery*: Packet delivery is solely based on the destination address and there is no source address validation by default. Thus, source IP addresses can easily be misattributed, or spoofed, which is a primary feature used by volumetric reflective

amplification DDoS attacks [OSZ20].[3]

- *Limited Expressiveness in TCP/IP protocol stack*: IP addresses, even with transport port numbers, cannot expressively describe the semantic characteristics of application-layer traffic, which makes it difficult for DDoS defense mechanisms to inspect traffic to identify attack packets.

Moreover, [HG04] proposed that a DDoS resilience architectural would need: (i) limiting the access to a server based on the server's capabilities, (ii) source address authentication to prevent source address spoofing, (iii) separating client and server address space to prevent unwanted traffic from client to client and server to server, and (iv) building symmetric traffic flows to prevent reflection attacks at the network layer.

### 4.2.3 DDoS in NDN

In NDN, attackers can only attempt to DDoS a target by flooding it with Interests because attackers cannot actively send Data packets to a target if the target has not sent corresponding Interest packets beforehand. Specifically, inspired by the work [GTU13], we categorize Interest packets used in DDoS attacks into three types according to (i) whether the Interest is valid, and if valid, (ii) whether or not the data requested by the Interest is dynamic (i.e., generated upon the Interest).

**Valid Interest for static data (Valid-S)** Valid-S Interests fetches Data packets that can be cached, e.g., Data packets for a CSS file or a video chunk. In NDN, Valid-S attacks can be mitigated because multiple Interests asking for the same data can be aggregated and satisfied by the in-network cache. NDN's intrinsic mitigation is sufficient unless attackers flood a huge amount of Interests traffic from a large spectrum of names.

[3]This type of DDoS attack has resulted in the largest attacks seen on the Internet to date.

**Valid Interests for dynamic data (Valid-D)** Valid-D Interests request data that is dynamically generated by producers upon the arrival of the Interest packets, for example, Interest packets used in a remote procedure call. This is usually reflected by a dynamic and unique data name carried by Interest packets. Since the Interest's name is customized and the Data is generated in real time, hardly any Interests arriving at a forwarder would hit cache or an existing Interest with the same name.

**Invalid Interests (Invalid)** An invalid Interest packet will not fetch an Data packet back because of its unrecognized format, unverifiable signature, incorrect application-layer content, etc. This type of Interest is mostly useful to malicious adversaries because legitimate applications can generate correct Interests following certain naming conventions. A possible way to generate such Interests is to append non-existent name components (*e.g.*, randomly-generated garbled bytes) to valid server prefixes.

When Valid-D and Invalid Interests are used in a DDoS attack, since their names are arbitrary and can hardly be satisfied by the cache, additional DDoS mitigation services are needed over NDN.

### 4.2.4 Design of FITT

We demonstrate by a new DDoS mitigation solution over NDN, Fine-grained Interest Traffic Throttling or FITT, that NDN's architectural changes, even when incrementally deployed, can make DDoS attacks fundamentally more difficult to launch and less effective. FITT leverages the NDN design to enable the network to detect DDoS from victim's feedback, throttles DDoS traffic by reverse its exact paths through the network, and enforces control over the misbehaving entities at their sources.

In a nutshell, FITT reacts to feedback from a victim, traces back the specified traffic flows by checking NDN's forwarding state, and enforces traffic throttling at the edge. The design is enabled by two major architectural properties of NDN. (i) The named traffic allows an

expressive way for a victim to specify the attacking traffic and for FITT to make fine-grained throttling. (ii) The stateful forwarding provides run time insights into ongoing traffic flows so that FITT can trace the traffic to precisely identify attackers.

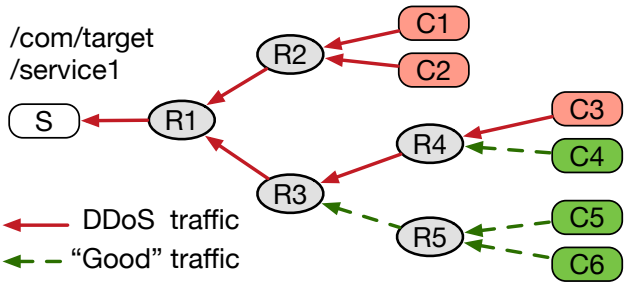



Figure 4.8: An example topology of NDN Interest flooding attack

FITT works in three main steps as shown in Figure 4.9. We illustrate the three steps with the example topology in Figure 4.8,

**Step 1: Detection** When server $S$'s service receives traffic more than the configured threshold, it sends out the feedback to its downstream router $R1$. Once receiving the feedback, $R1$ parses the feedback and triggers the FITT reaction based on the attack type.

**Step 2: Traceback** In order to identify attacking sources, $R1$ checks its forwarding state. Using names carried in the feedback from $S$, $R1$ can identify the attacking traffic flows by their names and then know the downstream routers ($R2$ and $R3$) from which the traffic is from. $R1$ notifies the downstream routers and $R2$ and $R3$ will perform similar procedures as $R1$ does. In this way, FITT reversely traces the attack traffic from $S$ all the way to *edge routers* $R2$ and $R4$ where exact traffic senders are connected.

**Step 3: Throttling** The edge routers will first notify these clients and then perform Interest throttling on suspect downstream interfaces within the specific prefix reported by $S$. During the throttling, an edge router will check whether a client has changed its behavior

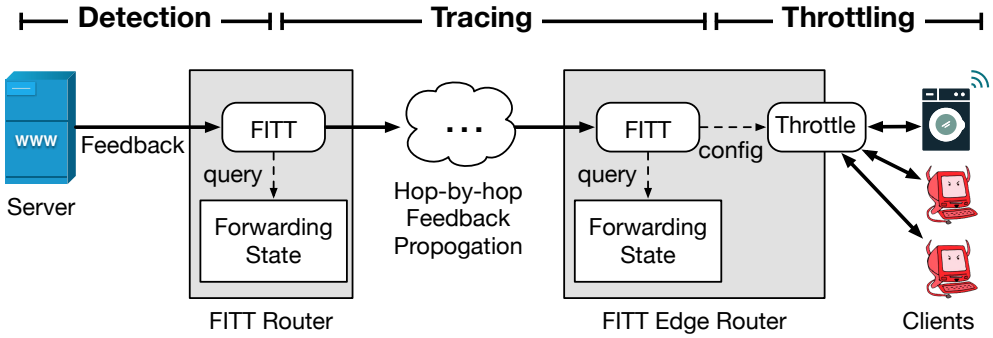



Figure 4.9: An overview of FITT system

or not (i.e. whether it lowers down its sending rate to the required value under the specified prefix). The router can then relax or reinforce the limit, accordingly.

Multiple FITT reactions can be triggered at the same time for different traffic prefixes (located on the same server or different servers) and different types of attacks. On the edge, if traffic for a specific prefix from a suspect downstream interface is being throttled by multiple FITT instances, a minimum allowed traffic value will be taken.

Since the mitigation is triggered by the victim's feedback, FITT can react immediately after the DDoS attack. The latency for throttling to start is only a one-way trip time (0.5 RTT) from the victim server to clients. In addition, since the whole process entirely operates over the existing NDN forwarding plane, there is no man-in-the-loop for DDoS mitigation with FITT.

### 4.2.5 Incremental Deployment

We further demonstrate that service providers may implement NDN/FITT on existing CDN nodes as an incrementally deployable solution to effectuate the application level remediation at the sources, which remains unattainable in today's DDoS mitigation approaches.

In this paper, we observe that the suitability of NDN's architecture to perform DDoS remediation is not just a parallel benefit to its suitability to performing CDN functions;

rather, we posit that existing CDN deployments are opportune infrastructure to enable broad deployment of NDN. Importantly, many CDNs' existing roles as MaaS providers suggest the potential alignment of costs with incentives to performing both CDN and MaaS.

- First, an upgrade of a CDN to support NDN could provide a synergistic benefit to Internet services that want protection from DDoS, and to the CDN/MaaS provider. Multiple aspects of the synergy that exists between DDoS mitigation and NDN. For example, one of the devastating aspects of volumetric DDoS attacks occurs when attack traffic is backhauled from distributed sources towards destinations (whether the destinations are victim services, or even to MaaS scrubbing centers) [OSZ20]. Deployment of NDN across a CDN/MaaS would let that provider shed attack traffic at the edges before it starts to aggregate across transit links.
- Second, CDN/MaaS providers already shoulder the computation and network requirements that NDN would require. For the TCP/IP Internet, CDN/MaaS providers have already been terminating TLS connections and proxying connections from clients to service infrastructures. An NDN upgrade would fit the operational footprint that large CDNs already have, would not necessitate the deployment of additional resources, but would also fundamentally enhance DDoS service offerings.
- Furthermore, among the properties of NDN's data-centric architecture is its inherent capability to cache data near its consumers, architecturally. This fundamental advantage is poised to not only be a pivotal feature, but also a deployment incentive for large Content Delivery Networks (CDNs) providers. In today's TCP/IP Internet, CDNs are vast networks and deployments that operate as overlay services for end-users. CDNs exist in TCP/IP networking (above the architectural layer) to efficiently deliver content to users today. CDN services typically involve large network infrastructures and deployments. They often perform caching of content to locations that are geographically distributed, as well as geographical load-balancing of requests from clients so that network latency can be minimized. In today's Internet CDNs exist as overlay technologies, and some

have proposed that NDN is well suited to implement these functions in the network architecture, itself [CPZ16, JB14, MCC14]. Furthermore, many operational CDNs, today, also perform DDoS mitigation offering commercial DDoS/MaaS services: Akamai [Aka19], Cloudflare [Clo19], and Neustar [Neu19] to name a few.

The resulting NDN network would have global scope and in-network caching in the topologically distributed regions. In general, incremental deployment models become more realistic when they align their costs with incentives. That is, those who deploy new mechanisms are more likely to do so when they anticipate direct benefits from doing so. By contrast, deployments like ingress and egress filtering (BCP-38 [FS00] and BCP-84 [BS04]) illustrate slow adoption, arguably, because those deploying them do not gain any direct benefits. Conversely, service providers already expend resources and money to combat DDoS by either provisioning large amounts of excess bandwidth or by contracting with commercial DDoS mitigation providers [Aka19, Neu19, Clo19]. Service providers whose applications may already be in a position to benefit from migrating to NDN's architecture would gain *additional* benefits by deploying NDN with FITT and thereby being able to shed large amounts of DDoS traffic.

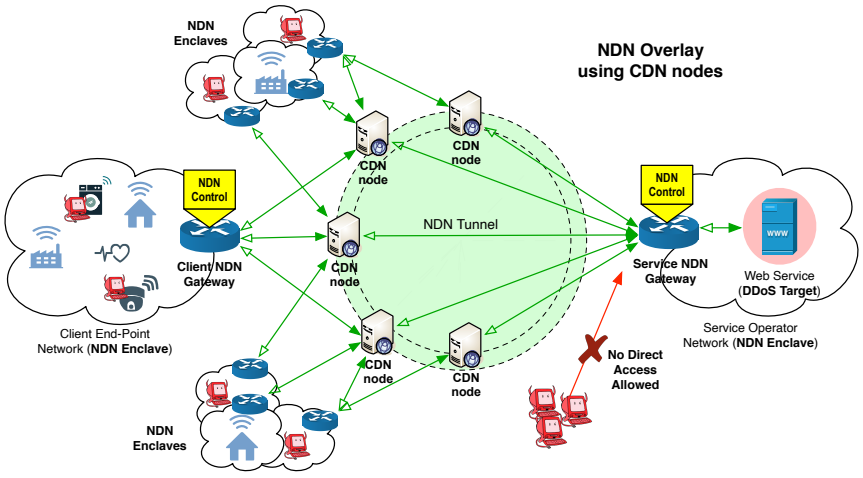



Figure 4.10: Incrementally deploy NDN and FITT over today's CDN

Therefore, we propose an incremental deployment (see Figure 4.10) that leverages CDN

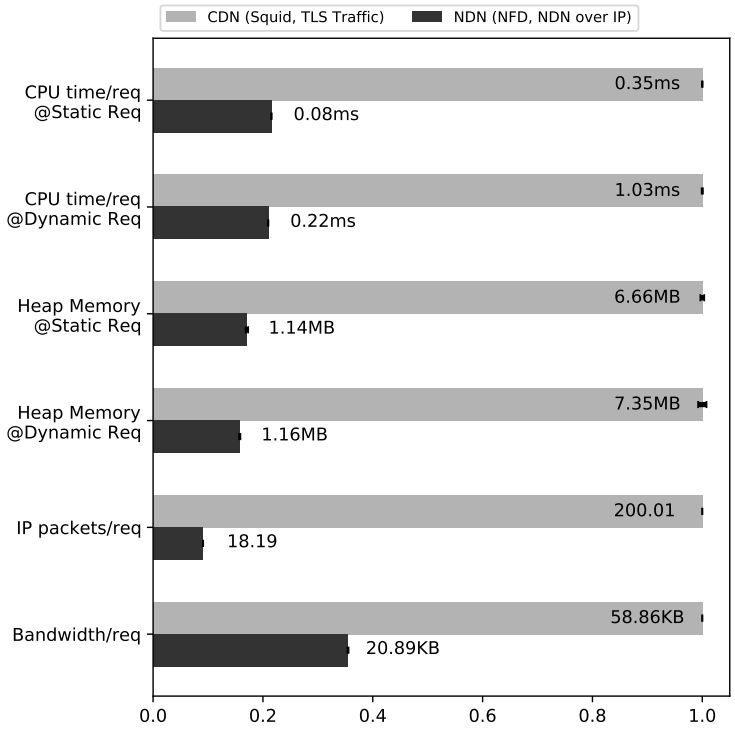
to connect NDN routers located at the edges of the network. The two ends speaking NDN are enough for FITT to effectuate the DDoS mitigation. This means that our approach requires NDN gateways on both sides of the CDN overlay: the client end-points and the service provide/operator side. On the client end-point, the NDN gateway can both forward traffic from existing NDN devices and translate at the application level using an application proxy traffic from traditional TCP/IP devices. Such a translation is akin to having an application proxy (i.e. web proxy) and it would allow the client NDN gateway to perform enforcement without having to worry about packet classification or deep packet inspection. On the service operator side, the NDN gateway can serve traffic directly to Web Service. What's more, service operators who migrate their services to NDN bolster each others' NDN deployments, as those clients independently augment each others' deployments (through facilities like shared caching and shared routing infrastructure). In particular, we observe that serendipitous IoT deployments of NDN, which may already be underway, could benefit other services whose providers have (or will) independently embraced NDN for this reason. That is, an NDN-enabled service may shed DDoS traffic from would-be attack nodes that might otherwise be bots in Mirai. By enabling NDN at the edges in home routers and IoT deployments would place the FITT mitigation machinery very near to some of the Internet's most voluminous DDoS sources for all NDN applications (not just IoT). We believe that it is demonstrably feasible for independent service operators to overcome network protocol ossification and migrate (at least portions) of their production traffic to NDN. Furthermore, we show later in this section that the FITT/NDN deployment outperforms existing CDN caching schemes because it is performed as a network function deeper into the stack. Thus, by enabling NDN at the edges, we get both performance and security benefits without sacrificing functionality.

The deployment of NDN/FITT may also not need to bother the change of existing end-point applications running in NDN enclaves. To be more specific, the gateway of NDN enclaves can be an NDN forward/reverse proxy. For clients, e.g., IoT devices, to talk to

a remote service, the client's gateway server plays the role of a forward proxy which turns application-layer request (e.g., HTTP GET request) into an NDN Interest packet and sends it out to the server over the NDN tunnel; while for the service provider side, the server gateway router serves as a reverse proxy parsing NDN Interest packets back to normal request. When servers send back the response, the process is similar. Given the commonalities between NDN's Interest-Data exchange and today's widely-used request-response model in application layer protocols (e.g., HTTP, RPC), such proxy and reverse proxy are deployable and the cost can be reasonable because it does not require any hardware change – all NDN proxy deployment is software installation in user space.

**Performance Comparison of FITT vs CDNs** We simulated a topology similar to Figure 4.10 where a CDN infrastructure is used for static and dynamic page delivery for web services. In the pure CDN scenario, the CDN proxies are running the latest version of Squid [Squ] and for our approach, the same CDN proxies run a version of our FITT prototype on top of NDN. Our aim was to be able to evaluate the computation and communication overhead of NDN forwarding daemon (NFD), an open-source NDN network forwarder, and Squid, one of the most widely used web proxy systems. Specifically, we performed multiple simulations using today's practice of MaaS and NDN/FITT's DDoS mitigation with the same hardware settings. In our experiments, we used computers equipped with Intel Core i9 4.6GHz processor with 32GB DDR4 RAM. We also used the same number of clients requesting the same amount of data and computation: six (6) clients requesting both static content and dynamically-computed content of 2KB at the same rate simultaneously.

As shown in Figure 4.11, we first present the CPU and memory use of a single-thread Squid and NFD/FITT under the same load. The plot indicates that the combination of NFD with FITT has an advantage in terms of resource consumption when compared to Squid: under the same load of traffic, Squid consumes an estimated of about five times (5x) more memory than FITT. We observed a similar trend when we measured the computation




(i) Heap memory consumption for static and dynamic requests (static requests will be satisfied by both CDN cache and NDN cache while dynamic request will be forwarded by CDN/NDN proxy).

(ii) CPU consumption for static and dynamic requests.

(iii) Number of packets and traffic received (TLS over traditional TCP/IP versus FITT tunneling NDN packets from an NDN enclave)

Figure 4.11: Comparing a vanilla CDN proxy running Squid and the same proxy running a prototype of FITT on top of NDN

footprint on each of the CDN proxies: FITT requires a mere 20% of the processing time when compared to Squid. In addition, we observed the total amount of traffic at the networking layer (IP) involved assuming a CDN proxy running Squid receiving TLS traffic vs NDN traffic (similar to Figure 4.10. In both cases, we made the assumption that traffic is forwarded over a single hop between the CDN and the MaaS scrubbing center. This is the worst case scenario for us because, in practice, MaaS scrubbing centers can be deeper into the network in a centralized location and several network hops away from the edge CDN proxies. Even under that assumption, our experiments indicate that in order to fetch the same amount

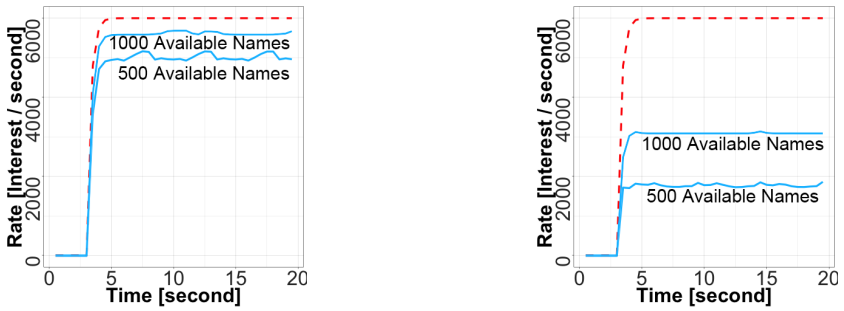
of content and computation results, Squid over TLS requires around ten times (10x) the amount of packets resulting in almost three times (2.8x) bandwidth overhead compared with NFD/FITT over IP-overlaid NDN. As we mentioned, since the MaaS service requires TLS-terminating traffic forwarding on both the CDN proxies and MaaS scrubbing centers, the computation and communication overhead can be much higher than what we report here for the vanilla CDN implementation making FITT a much more desirable option.

### 4.2.6 Evaluation

Besides the prototype implementation over the latest stable version of NFD, we also implement FITT in C++ over ndnSIM [MAZ17], which is an NDN simulation platform based on NS-3. We first demonstrate NDN's DDoS resilience to valid static Interest flooding and then evaluate FITT under different types of attacks. The simulation results show that after the DDoS starts, FITT can effectively control the traffic to the victim as expected within seconds (less than 2 seconds under our simulation settings), and ensure that over 99% of the attack target(s) incoming traffic is from legitimate clients after a short period of time.

In addition, we simulate the scenarios when (i) multiple DDoS mitigation instances are happening at the same time, (ii) different traffic flows are presented and only one of them is throttled, and (iii) attackers are rogue and follow the DDoS control of FITT. The results show that FITT can perform fine-grained DDoS mitigation and handle comprehensive attack scenarios.

**Simulation Topology** We simulate the incremental deployment of NDN/FITT using a CDN as an overlay between NDN enclaves. As shown in Figure 4.21, the blue nodes are CDN nodes that are aware of NDN and FITT, the gray nodes are gateway routers of NDN enclaves. Behind each gateway router, we simulate 10 compromised IoT devices and 2 honest IoT devices. Since FITT follows a divide-and-conquer strategy in traffic throttling, the scale of the network topology does not affect the evaluation results of FITT much. We make the




(a) Interest Aggregation (b) In-network Cache

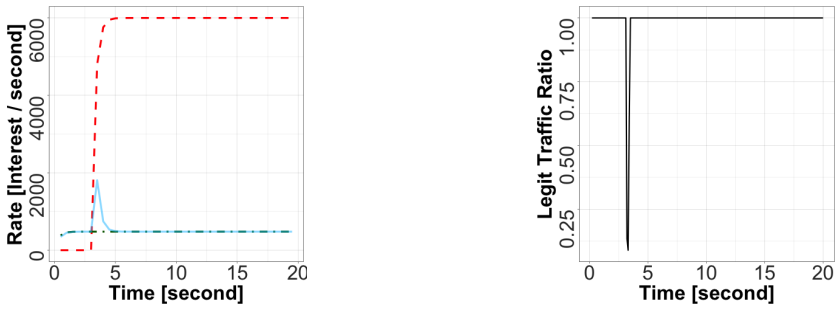
**Simulation Settings:** NDN deployment without FITT. Each attacker (60 in total) starts sending Valid-S attacking Interests at 100 pkt/s from the second 3 with an available number of data names 500 and 1000, respectively.

Figure 4.12: NDN's DDoS Resilience to Valid-S Interest Attack

service globally reachable, which indicates all users have the means to learn the name and express Interest packets towards the service. For sake of simplicity, we use the prefix $P$ to represent the service in the rest of the section.

**Simulation Result Notation** In the simulation result plots, we use the red dashed line to represent attackers' sending rate (RPS), the blue solid line to represent DDoS target's receiving traffic rate (RPS), the green dot-dashed line to represent legitimate clients' sending rate (RPS). Therefore, when the blue solid line meets the green dot-dashed line, all traffic arriving at the DDoS target is from legitimate clients. The area below the red dashed line and above the blue solid line represent the DDoS mitigation provided by FITT over NDN.

**NDN: Resilience to Valid-S Interest Flooding** Figure 4.12 demonstrates NDN's DDoS resilience to static (Valid-S) Interest flooding with NDN's intrinsic properties, i.e., Interest aggregation and in-network cache. To be more specific, we first disabled cache in all routers so that the result will only be affected by NDN's Interest aggregation. As shown in Figure 4.12a, NDN can withhold traffic from attackers (red dotted line) to the server (the blue solid line). The more available names are used, the less traffic NDN can retain. We then




(a) Invalid Interest DDoS (b) Legitimate Traffic Ratio

**Simulation Settings:** NDN deployment with FITT. Each attacker (60 in total) starts sending invalid Interests at 100 pkt/s from the second 3. Legitimate clients' sending rate is 40 pkt/s starting from the second 0.

Figure 4.13: FITT mitigation of invalid Interest attack

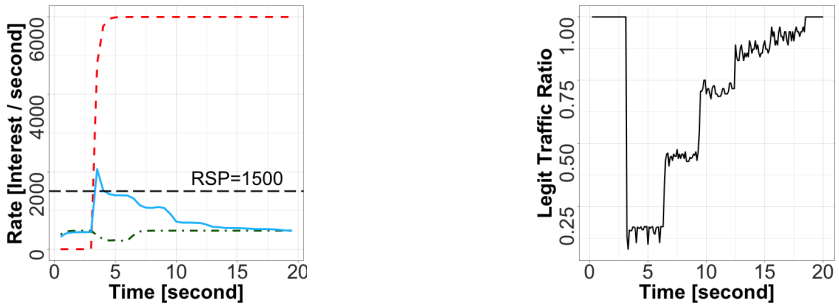
introduce the cache capacity of 200 data packets in Figure 4.12b. It is apparent that the number of Interests reaching $P$ decreases because of the caching capacity, which is because intermediate nodes along the path will serve future same Interests with cached Data (the freshness of cached Data is 4 seconds in our simulation).

The two figures indicate that the effect of Interest aggregation and cache is lower when an attacker can use a bigger set of Interest names to attack the victim. This is because the larger the name set, the smaller the chance of two Interests carrying the same name, and the smaller the chance to hit a previously cached Data packet.

**FITT: Invalid Interest Attack** We first study FITT's performance against Invalid Interest (Invalid) DDoS attack. As shown in Figure 4.13a, initially, $P$ only receives Interests from legitimate clients (green dot-dashed line). After 3 seconds, attackers start the DDoS by sending Invalid Interests (the red dashed line) to $P$. As depicted by the plot, $P$'s incoming traffic line (blue solid line) immediately goes up after the attack but soon goes down and merge the legitimate clients' outgoing traffic line (green dotted line). Therefore, FITT can eliminate the invalid Interest DDoS traffic in a short time. The effectiveness is because in invalid Interest DDoS attacks, FITT can accurately identify attackers by *InvalidNames*

carried in feedback messages and throttle their attack traffic. Figure 4.13b shows that after FITT reaction, all the traffic received by $P$ is from legitimate clients.

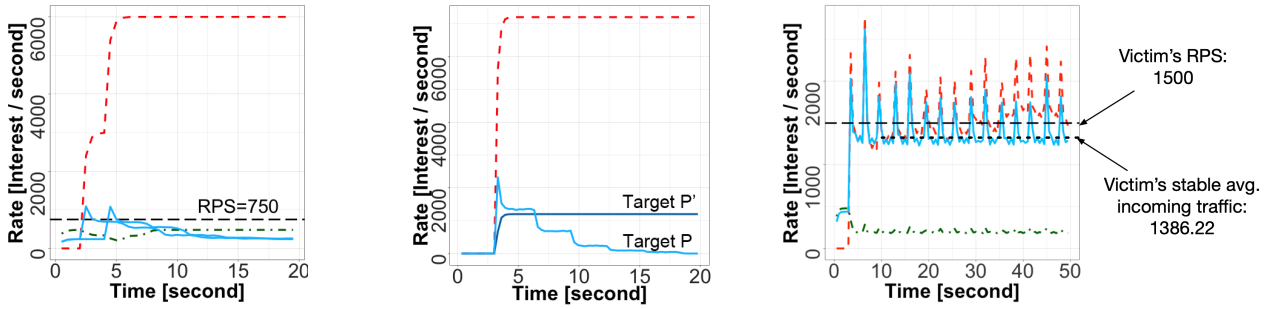
**FITT: Valid Interest Flooding** We then simulate valid Interest flooding. To remove the effect of in-network cache, we disabled all router's cache. As such, Valid-S and Valid-D Interest flooding become the same because all of them will arrive at the DDoS target $P$. The results of Valid-D Interest flooding are shown in Figure 4.14.




(a) Valid-D Interest DDoS (b) Legitimate Traffic Ratio

**Simulation Settings:** NDN deployment with FITT. Each attacker (60 in total) starts sending Valid-D Interests at 100 pkt/s from the second 3. Legitimate clients' sending rate is 40 pkt/s starting from the second 0. We let $P$'s capacity be 1.5K RPS and *RateLimitTimer* be 3 seconds.

Figure 4.14: FITT mitigation of valid Interest flooding




(a) Two-victim DDoS attack (b) Fine-grained throttling (c) "Smart" attackers

Figure 4.15: FITT mitigation in different attack scenarios

Since the router cannot tell good traffic from bad traffic when DDoS starts, legitimate clients are also limited. After receiving the FITT feedback messages, legitimate clients will

abide by the control placed and lower down their sending rate until the router determines them to be legitimate and free the limits, explaining why the green dot-dashed line goes down in the first several seconds of the attack and then back to the normal later. As for attackers, as shown, the traffic received by the victim drops periodically (every 3 seconds), which confirms the FITT's reinforcement throttling: FITT will halve the limit on attackers until all the attackers' traffic to the reported prefix are totally blocked. At the end of the mitigation, >99% of the Interests received by the victim are from legitimate clients (Figure 4.14b).

**FITT: Multiple Attacks to Different Prefixes** FITT is designed not only to handle a single DDoS attack, but also comprehensive DDoS attacks, *i.e.*, attacks to different prefixes, starting at a different time and using different types of Interests. In this simulation, we evaluate a more complicated attack scenario where half of the attackers attack service $P$ with Valid-D Interests starting from second 2 and another half attacks service $P'$ with Valid-S starting from second 4. We found the simulation results of scenarios when $P$ and $P'$ are located on the same node or different nodes are almost the same. Figure 4.15a shows the result when $P$ and $P'$ are running on the same server.

As shown, when multiple FITT mitigation instances take place, FITT can effectively control the DDoS traffic from both attacks at the same time. For each victim server, the incoming Interests are throttled in a similar way like that when there is only one victim server under attack. Two servers' incoming traffic lines quickly go below the threshold after the attack started and soon merge the legitimate client traffic line, indicating that all the traffic received by the two servers is from legitimate clients.

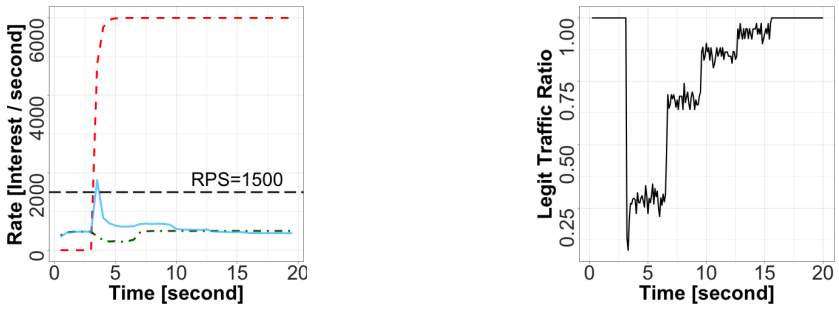
**FITT: Throttling Granularity** In DDoS mitigation, collateral damage may ruin the legitimate traffic sent from the compromised devices. FITT throttles Interest traffic at a granularity of flow under a specific name prefix. We evaluate FITT in terms of the granularity

of the traffic throttling. We reuse the simulation settings of the two-victim scenario to let all the attackers not only attack $P$ with Valid-D Interests at 100 pkt/s but also keep the normal communication with another service provider $P'$ at the reasonable rate of 20 Interests/s. In this case, the attacking traffic will overwhelm $P$ but the legitimate traffic will not go beyond the capacity of $P'$. As shown in Figure 4.15b, compared with the two-victim scenario where traffic to both $P$ and $P'$ will be throttled, in this simulation, FITT only squelches clients' traffic under $P$ while the traffic towards $P'$ will not be affected.

**FITT: "Smart" Attackers** Attackers may try to circumvent FITT's traffic throttling by complying feedback messages from edge routers temporarily and switching back to attack mode later. However, when attacks switch back to a high sending rate, the traffic volume will alarm the service's pre-configured threshold again, which will in turn force the attackers to lower down the Interest sending rate again. In this way, the FITT will be triggered periodically when switches take place (Figure 4.15c). Consequently, the average attacking traffic will be kept at a certain level where the service will not be spoiled, especially when the threshold is properly set below the real capacity of the service.

**FITT: Mixed Interest Attacks** We tune the attackers in the Valid Interest flooding scenario to send both Valid-D and Invalid Interest packets to simulate a mixed Interest attack.

As shown in Figure 4.16, compared with results of Valid-D Interest flooding scenario, one obvious difference is that, after the attack starts, FITT will drop the traffic to be much lower than $P$'s $RPS$ (black horizontal line). This is because, at the edge, an invalid Interest attack will lead to a total block of the attackers, which cancels out the $RPS$ from the valid Interest attack mitigation. After a short period, FITT will place the limit to misbehaving clients only and the legitimate clients will recover. In the end, FITT will only pass legitimate traffic to $P$ (Figure 4.16b).




(a) Mixed Interest DDoS (b) Legitimate Traffic Ratio

**Simulation Settings:** NDN deployment with FITT. Each attacker (60 in total) starts sending both Invalid and Valid-D Interests at 100 pkt/s from the second 3. Legitimate clients' sending rate is 40 pkt/s starting from the second 0. We let $P$'s capacity be 1.5K RPS and *RateLimitTimer* be 3 seconds.

Figure 4.16: FITT mitigation of mixed Interest flooding

## 4.3 DLedger: Distributed Immutable Logging

The majority of the materials contained in this section were modified or quoted verbatim from [ZVM19].

DLedger is an NDN based permissioned distributed ledger. The main idea of DLedger to make each record a piece of named secured data in the permissioned The records will be directly used in the NDN based peer-to-peer network as the network layer packets, thus allowing a coherent network and application design.

### 4.3.1 Background and Motivation

The open access and lack of trustworthiness among peers in public or permissionless blockchain do not apply to permissioned application systems where strong trust relationships need to be defined and enforced among a group of entities. For example, in a smart grid network system, the participants like customers, device manufactures, power plants, public utilities, and government agencies [The14] do not fully trust each other but share the same goal of providing reliable power support. Under the same goal, each recorded data (*e.g.* logs of executed or pending commands) should be strictly validated before it can be confirmed and

accepted by the system, *e.g.*, the data generator is authorized against the policies and the content is correct against some out-of-band or online query. Such a permissioned system should also provide high throughput, low latency, and fault tolerance for application use.

The state-of-the-art permissioned blockchain system, Hyperledger Fabric [ABB18], improves throughput and reduce latency by letting each transaction be executed only by a subset of the peers and thus parallel execution. Nevertheless, the writing of each block into the chain still remains a single head-of-link blocking process because Fabric uses one global blockchain and employs a fully-synchronized, conceptually-centralized ordering service to establish the order of transactions. In addition, Fabric uses TLS as the underlying communication channel for gossip and thus defeat the asynchronous communication and subject to network partition.

Our work is inspired by an experimental private solar system where the customer energy production/consumption records measured by the rooftop solar gateways were kept on a cloud server for logging. These records will later be quoted by the business service provider to bill its customers. However, since a centralized ledger is governed solely by the service provider, this approach is often maligned by customers as well as other financial parties involved in business systems for the following two reasons. First, the business service provider who controls the server may tamper customer records to obtain unfair advantage. For example, the solar system service provider may alter the energy usage data to maliciously charge customers who cannot prove the existence of original recording data. Even worse, today, there is little surveillance of such corruption. Second, cloud server outages and unstable IoT network connectivity could also probably ruin the data availability.

### 4.3.2 Design of DLedger

We describe DLedger, an open-source permissioned distribute ledger system over directed acyclic graph (DAG) data structure and Named Data Networking (NDN) as a secure and asynchronous peer-to-peer (P2P) network. DLedger adopts a number of new concepts and

techniques:

- Parallel block writing and hard fork avoidance allowed by the DAG.
- Reference based endorsement allows blocks to be confirmed when a subset of the peers endorse it according to the policy. This is similar to Fabric's idea but differs in the following two aspects. First, each block is endorsed differently per pre-defined policies. For example, more critical logs require a larger number of endorsements or endorsements from certain entities. Second, the endorsements themselves are also recorded by the DLedger for auditing purpose.
- Co-design of communication layer using NDN and data layer using named secured block. DLedger let each block a signed and named NDN Data packet. Each block is a node in the DAG data structure in the data layer and is a self-containing unit that can be cached, fetched, and verified in the communication layer.
- No special centralized functions.
- A number of endorsement policies to improve security of the system and prevent lazy peers.
- Support of distributed programs (like smart contracts used in Ethereum) written in standard programming language with Web Assembly Interface (WASI).

Our design favors availability (*i.e.*, liveness) over consistency (*i.e.*, safety) and provide eventual consistency considering possible network partitions. Specifically, when a network partition happens, critical blocks can be written into the system without any hard forks, but depending on the endorsement requirement of different type of blocks, they may not be confirmed until the networks rejoin.

**Communication Layer** Entities utilizes Named Data Networking (NDN) for data dissemination and synchronization. All entities in a DLedger system, including the identity

manager, together form a peer-to-peer (P2P) network. Every record are generated with a name, and other peers can fetch this data from NDN network directly using the name.

Entities advertise latest records to the whole P2P network. This process triggers both periodically and after out-of-sync events (e.g., new record generation, sleeping mode, network failures, etc.). After receiving the synchronization Interest packets, entities will fetch the missing records and add it to their local ledger after record validation. Leveraging the synchronization protocol, DAG is replicated and synchronized among all entities.

The benefits of using NDN as the P2P network brings a number of benefits:

- Being an NDN Data packet, each block can be fetched, cached, verified from any peer in an asynchronous manner. This is entirely handled by the NDN protocol stack and thus there is no need to design and implement it in the application logic (*e.g.*, by building a distributed hash table to keep a block-entity mapping and implementing a open-access database of blocks on each node).
- NDN allows efficient Data multicast. When spreading a new block to the entire network, conventional gossip network will cause a large number of duplicate transmission. In NDN, this is done in a request-and-response way where only lightweight Interest may be duplicatedly transmitted and the heavy block will be shared without any bandwidth waste.

**Data Layer** DLedger stores all records in a Directed Acyclic Graph (DAG) instead of a single blockchain. Each vertex represents a record carrying its payload data while each edge represents an *approval* made by a record to another. In DLedger's DAG, a newly generated record (vertex) will be placed adjacent (chained) to $n$ ($n \geq 2$) existing records in the DAG, establishing the approvals from the new record to the previous ones. *Tailing* records are those records not approved by any other records in the DAG.

When a peer $P$ generates a new record $R$, the peer takes the steps as follows.

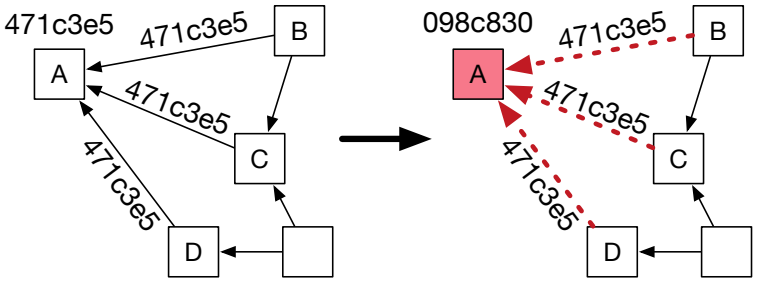


Figure 4.17: Endorsement ensures immutability in ReLiShare

1. $P$ randomly selects $n$ tailing records generated by other peers.
2. It will then verify the validity of these records and the records directly or indirectly approved by these records. If any of the $n$ records is invalid, i.e. the record is malicious or it approves an invalid record, $P$ should drop it and repick another.
3. The peer then adds the names of the $n$ selected records and the application payload into the new record $R$ and sets $R$'s name to:

   /DLedger/<Generator ID>/<Record Hash>

   where <Generator ID> is $P$'s unique ID and <Record Hash> is the digest calculated over $R$. This will ensures the immutability of the blocks being referred by the current block because a modification of payload will lead to broken references in later blocks (Figure 4.17).
4. Finally, $P$ appends a digital signature for block authenticity and integrity.

Each peer appends its new records into the DLedger after making *approvals* to other peers' records whose validity can be successfully verified. DLedger adopts a set of security policies for record receivers to check the incoming records, preventing potential threats such as spam attack and collusion attack, and misbehaving operations like laziness (contributing nothing to system security).

Specifically, besides the verification of packet format, the content, signature, and block creator's authorization, DLedger also has the following rules.

- Interlock Rule: Creator may only attach new blocks to blocks generated by other

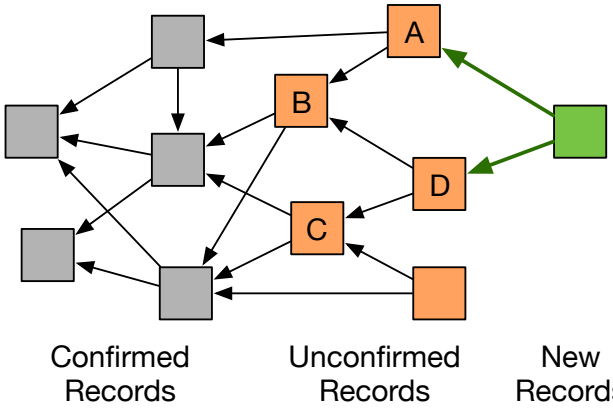


Figure 4.18: DAG, endorsement, and consensus in DLedger

entities. This prevent a peer from maintaining a secret chain or DAG and greatly increasing the verification time of blocks (because a verifier needs to verify a entire sub DAG or chain.)

- Generation Rate Policy: Creator must generate block below a certain rate to prevent flooding.
- Contribution Rule: Creator may only attach new records to relatively new unconfirmed records. This prevent lazy peers from skipping verification of unconfirmed records.

**Consensus Layer** The consensus on a record in DLedger is realized by enough number of endorsements made by a number of entities. The number of endorsements from *different entities* is called the *weight* of a record. As shown in Figure 4.18, the green block increases the weight of record $A$ and $D$, and indirectly increase the weight of $B$ and $C$. When a record gains sufficient number of endorsement according to the security policy, it is confirmed and accepted by the whole group. This assures a $(k, N)$ $k < N$ threshold scheme where unless more than $k$ peers of the total $N$ peers get compromised, the system remains secured, according to the trust model.

**Application Data Format Layer** The distributed ledger stores multiple types of record content:

- Application records, *e.g.*, the log of executed and pending commands to a smart grid system.
- Identity management operations which include certificate issuance and revocation.
- Security policy records.
- Executables (*i.e.*, smart contract) in the compiled binary format following WASI.

### 4.3.3 Evaluation

In this section, we evaluate DLedger's robustness, scalability, and ability of handling network partition through the theory analysis and our simulation results over ndnSIM [MAZ17], an NS-3 based simulation platform for NDN.

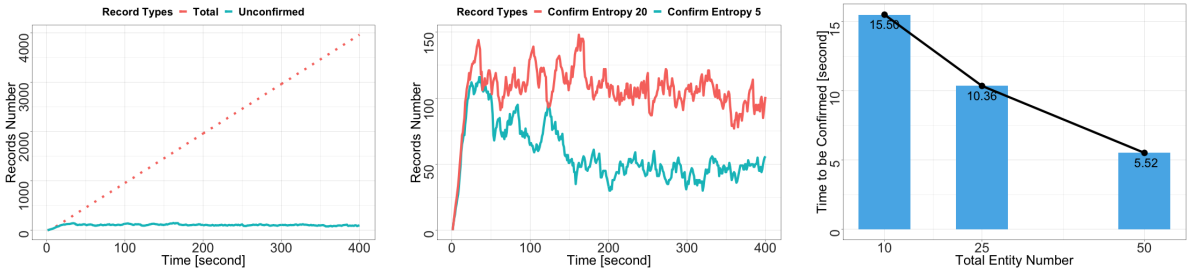
**The Unconfirmed Records Size** An essential concern of DLedger is its robustness: as system runs, will the size of unconfirmed records go infinitely large resulting in the system crash? As proved in [ZVM19], the tailing record size is concentrated around the constant:

$$\frac{n\lambda T}{n-1}$$

where $T$ is the average latency for a new record to be accepted by the system and $\lambda$ is the new record generation rate of the system following a Poisson distribution. Moreover, the proof in [ZVM19] shows that the time for a record being confirmed is also a constant value. Therefore, since both the frontier (tailing record) and the depth (time to be confirmed) of DAG's unconfirmed records keep constant, the unconfirmed record size is also a constant value.

We evaluate DLedger by the simulation in terms of the unconfirmed record size as the DAG grows. To be specific, we set up a 50 entities P2P network with $W_{confirm} = 20$,

where each entity's record generation rate is 0.2 records/sec. As shown in the Figure 4.19a, the unconfirmed record number will not increase with the DAG's growth. Instead, the unconfirmed record number line will increase for a short time after bootstrapping, and then fluctuate around a constant value, confirming the mathematics proof.




(a) (b) (c)

Simulation Settings: (a) (b)50 entities P2P network with $W_{confirm} = 5$ and $W_{confirm} = 20$. (c) Total entity number is 10, 25, and 50. Each entity generates a new record every 5 seconds.

Figure 4.19: Unconfirmed record number and confirmation time in DLedger

The Figure 4.19b reveals the relationship between the $W_{confirm}$ setting and the unconfirmed record size when the total entity number is the same: by adjusting the $W_{confirm}$ from 5 to 20, the unconfirmed record size will increase by only about 100%.

**System Scalability** We argue that DLedger has better scalability because of the following two reasons. First, DLedger leverages DAG to keep the records. Instead of allowing only one successful next block in the blockchain, DLedger accept multiple valid next records, thus allowing multiple entities contributing to the system in parallel. This gets rid of the computation waste and frees the limit on record processing speed. Second, PoA is used as the gating function. Unlike PoW which is highly CPU bound, PoA is much cheaper and efficient even for constrained IoT devices, greatly improving the system efficiency.

We also show DLedger's performance when the number of entities grows. As shown in Figure 4.19c, with fixed $W_{confirm}$, the more entities participate, the faster a record will be

confirmed. Because for any specific record, more entities means more efforts on approving it.

**Network Partition** We evaluate DLedger's performance in the case of network partition: in the simulation, we split the network into two independent subnets by taking down the links between them. The partition takes 100 seconds and two networks unit again after that. Figure 4.20 shows a DAG maintained by an entity at the end of the simulation. As shown, two branches have formed and then merged, showing DLedger's ability to recover from the network partition, eventually confirming records from each subnet.

Note that each branch of the partition needs to have more peers than confirmation weight for this branch to be confirm and recognized by other branch; this prevents entities collude and flood bad records using a branch.

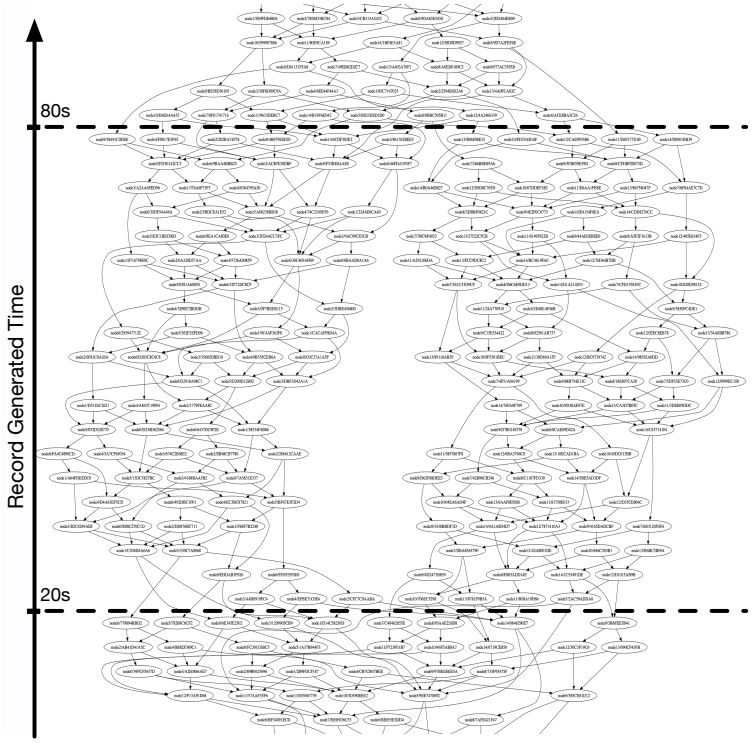
## 4.4 RapidVFetch: Secured Data Fetching in Vehicular Network

The majority of the materials contained in this section were modified or quoted verbatim from [ZLD19].

RapidVFetch is an NDN based vehicular data prefetch mechanisms. The main idea is to let the road side units (RSU) download the data desired by the future vehicles as NDN named secured data and cache it, so that when future vehicles arrive, it can directly fetch the data without the round trip to the original data producers or repositories. To facilitate the notification of desired data from vehicles to the RSU before the vehicles arrive at the RSU's coverage, we also utilize the NDN based vehicle to vehicle (V2V) communication.

### 4.4.1 Background and Motivation

In real world, due to the cost of deployment and support, RSUs can be deployed in many different environments with various densities and topologies [RST11].




This shows a DAG from the simulation. Each circle in the figure is a record. The links among records are the approvals. The recent records are higher in the DAG as shown in the figure.
Simulation Settings: 15 entities P2P network. The partition takes place at second 20, dividing the network into a 7-node subnet and a 8-node subnet. Two networks rejoin at 80 seconds.

Figure 4.20: Simulation of network partition in DLedger

In this paper, we target the problem of degraded data downloading performance for vehicles in areas with limited RSU coverage. To illustrate our problem better, we use a simplified example scenario; that is, a vehicle drives out of a previous RSU's coverage, loses the RSU connectivity, and then drives into the next RSU's range (Figure 4.21). In addition, we separate the target applications into two categories. For non real-time applications, the problem is how to improve data downloading rate by fully utilizing the limited connectivity time. For real-time applications, the problem is how to ensure a consistent downloading rate for the user.

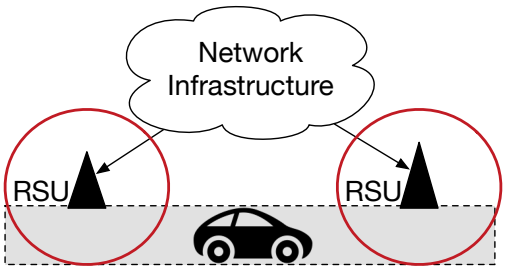



Figure 4.21: An example scenario of vehicular data prefetching

Content prefetching in mobile environments to reduce latency in data download has been well explored in a number of existing works [JK98, NN08, Cho02]. In the context of vehicular networking, due to the intermittent connectivity between vehicles and RSUs, prefetching of data content becomes vital for ensuring the quality of information services.

Prior works such as [DK09, GS18a, HZ16, KL15] calculate the possible trajectory of the vehicle and content popularity, then prefetch different chunks of content to RSUs to maximize data retrieval rate. Fundamentally, RSUs fetch the required content from the cloud server based on a prediction model, and vehicles fetch desired content from RSUs when moving inside their signal range. The prediction and caching strategy aims to facilitate this content fetching process with minimum delay.

These approaches were proposed to run over IP, but its design lean towards a data-centric solution as their proposed mechanisms make data independent of the IP's point-to-point channels. Following this direction, it becomes ideal to apply a data-centric solution to the vehicular content prefetching problem to avoid the overhead of address allocation and connection state maintenance.

A common issue with existing IP-based and NDN-based prefetching approaches is that their performance is dependent on using a centralized server for accurate predictions, as inaccurate estimation causes waste in storage resources, and hurts download performance. In this paper, we propose a requester-initiated prefetching solution to better satisfy the real-time needs of vehicles, and increase the utilization of in-network storage. Our solution does not make predictions based on predefined models, neither does it depend on prior travel paths

nor request history, but prefetches content on RSUs based on real-time user requirements in a timely manner.

An existing approach following a similar idea was [RZ13], which allowed vehicles to signal the currently connected RSU when a handover process was going to happen, and notify the following RSU to prefetch the required content. Therefore, the protocol is not transparent to RSUs. Other than [RZ13], existing approaches like [AB17] [GS18a] also make use of RSU and infrastructure side routers for content caching. This means RSUs and wired infrastructure side routers need software upgrades to support new protocols, which increases deployment difficulties. Rather than relying on RSUs or infrastructure routers to cache content, in RapidVFetch, we take an alternative approach by making use of V2V communication to initiate content prefetch requests and content delivery. In doing so, we do not need to upgrade existing RSU infrastructure, and only require the vehicles involved in the communication process to support our protocol. This also reduces the attack surface on RSUs as we can authenticate the communicating vehicles through physical connectivity to the RSUs.

Another issue with existing work is the lack of security considerations in prefetching. The prefetching process is asynchronous and how to ensure integrity, authenticity, and confidentiality of prefetched data becomes a main challenge. In RapidVFetch, we provide an approach to protect sensitive information in NDN names and Data content from eavesdropper (e.g., other vehicles, RSUs).

### 4.4.2 Design of RapidVFetch

We propose RapidVFetch to facilitate vehicular data downloading by prefetching named, secured data via V2V communication over NDN. Intuitively, since it is the vehicle who best knows its future locations and desired content, RapidVFetch lets each vehicle, called a *requester*, express their needs by Interest packets which are small in size and solicit help from other vehicles, called *forwarding vehicles*, through V2V communication. Vehicles at future

locations can simply express these small Interest packets to the RSUs, where the prefetched packets will be cached at the RSUs and will soon be used by the requester when it moves into the RSU's range. For real-time applications, the data can also be forwarded back to the requester.

RapidVFetch provides the following desirable features.

- Having each requester take care of their own needs, RapidVFetch provides accurate content prefetching at desirable RSUs without reliance on centralized services.
- By utilizing forwarding vehicles, RapidVFetch allows a requester to maintain stable data downloading for real-time applications.
- Enabled by NDN's built-in security, RapidVFetch ensures data authenticity and confidentiality regardless of the trustworthiness of other vehicles or RSUs involved in the downloading process.
- RapidVFetch is transparent to the infrastructure and RSUs, that is, RSUs and backbone routers do not need to be aware of RapidVFetch.

The goal of RapidVFetch is to facilitate and secure the data downloading from the infrastructure to vehicles by utilizing broadcast V2V communication and NDN's content-centric networking model. To be more specific, for non real-time applications, RapidVFetch aims to improve data downloading rate by fully utilizing limited connectivity time; for real-time applications, the goal is to keep the requester online and ensure a smooth downloading rate.

In a nutshell, as shown in Figure 4.22, a *requester* contacts a *forwarding vehicle* in the target RSU's range through V2V communication (which may contain single or multiple hops) to prefetch data (❶). For non real-time applications (e.g., video downloading), when the requester drives into that RSU's range, it can re-send Interests for the prefetched content and benefit from proactive caching (❷), improving utilization of the limited connectivity to RSU. In the case when real-time communication is required (e.g., online conferencing), the

forwarding vehicle can also forward the replied content back to the requester in order to keep the requester online (❸). Importantly, RapidVFetch considers each Data's security in the downloading process. The downloaded Data packets are signed and can also be encrypted using NDN's built-in security primitives and name-based access control (NAC) [ZYR18].

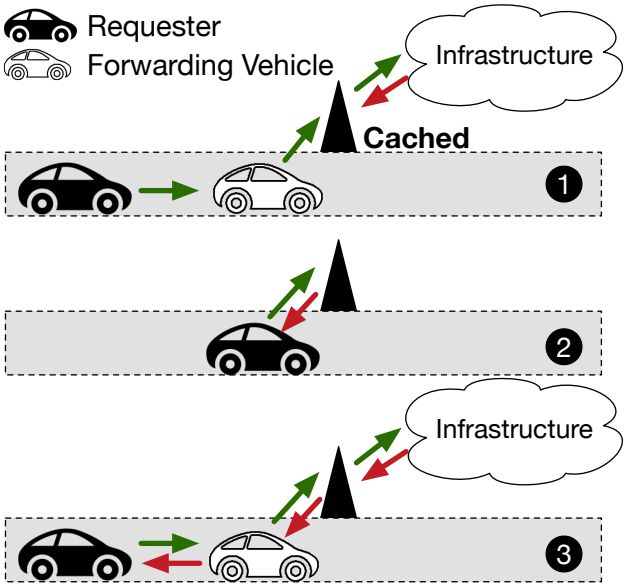



Figure 4.22: An overview of RapidVFetch

By letting each requester initiate their own prefetch Interest packets, RapidVFetch always provides accurate prefetched data at target RSUs, getting rid of the centralized service for content/location prediction. In addition, this allows the requester to have a better control on when to trigger the prefetching, so RapidVFetch has more flexibility to cope with complicated road conditions (e.g., unexpected delay). For example, if there is a traffic congestion before entering the next RSU, the requester can delay the prefetch. At the same time, since a RSU does not need to treat prefetching Interests differently, the RSUs do not need to be aware of the RapidVFetch protocol. Consequently, RapidVFetch can be deployed at vehicles only without bothering RSUs and the infrastructure behind RSUs.

Our work illustrates the superiority of using a data-centric network architecture in vehicular networking: NDN enables RapidVFetch to let vehicles themselves pick what to cache

in desired RSUs without reliance on a predictor; at the same time, the prefetched data is secured in a data-centric way and the whole process can be transparent to the infrastructure.

### 4.4.3 Securing Vehicular Data Prefetching and Downloading

In RapidVFetch, we consider the security of both the data prefetching process and the downloading process, which further includes (i) authentic use of RapidVFetch on RSUs, (ii) integrity and authenticity of downloaded data, and (iii) user privacy and the confidentiality of downloaded data.

**Authentic Use of RapidVFetch on RSUs** It is nontrivial to authenticate users in an open wireless network system. To date, a number of vehicular security architectures have been proposed to employ a Public Key Infrastructure (PKI) in vehicular networks. However, these approaches provide means to ensure a vehicle has a valid plate, the driver has a valid license, or the vehicle is made by a verified manufacture, but they are not sufficient to prove the application is honest, e.g., whether the application will send spam or DDoS traffic to a RSU. Therefore, in this work, to mitigate spam Interest packets towards the RSU, we consider a simple but reliable means to authenticate a user: only forwarding Interest packets sent by users within RSU coverage.

This feature also distinguishes our work from existing works. For example, [RZ13] proposes a solution where a previous RSU issues an Interest packet to the next RSU. However, it is difficult for a RSU to verify that the Interest is from another RSU and is truly originated by a vehicle that will later drive into its own coverage. In contrast, RapidVFetch reduces the attack surface and RSUs can keep the strict access policy (i.e., traffic within coverage) without being aware of the RapidVFetch protocol.

**Data Integrity and Authenticity** Acting as an Interest forwarder, the forwarding vehicle may direct the Interest to a fake service provider maliciously; as can the RSU, which

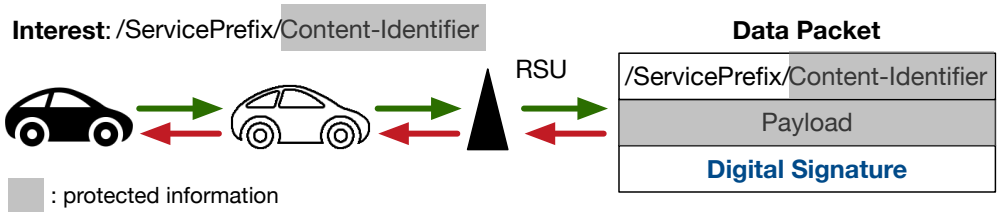
may alter replied Data packets in their cache. Such alteration or fake Data packets will degrade the downloading and may cause damage to the requester. It is difficult for today's channel-based security model (e.g., TLS, QUIC) to ensure the data integrity/authenticity because the data will be cached at the RSU. This is because by the time the data is used by the requester, the secured channel does not exist any more and thus no security can be guaranteed.

However, empowered by NDN's built-in security, in RapidVFetch, every Data packet is signed (Figure 4.23) by its original producer, e.g., the video streaming service provider. Consequently, such alteration or fake Data packets will be detected by verifying Data packet signatures. If the data has a bad signature, the requester should resend the Interest packets and fall back to the normal downloading mode.

**User Privacy and Data Confidentiality** A forwarding vehicle is able to know the Interest packets from the requester and the RSU can also grab information from the replied Data packets. This raises the user privacy concern for potentially leaking information of requester's ongoing actions and desired content. However, NDN names are not necessarily readable to eavesdroppers and the Data payload can also be protected through cryptographic techniques (Figure 4.23).

As a simple and effective solution to prevent information leakage, the requester and the service provider can negotiate a shared secret (e.g., through Diffie-Hellman protocol) asynchronously (e.g., ahead of vehicular communication). The shared secret can then be used to keyed hash or encrypt the sensitive name components in the Interest packets and content in the Data packets. For example, assuming the service provider's routable prefix is /service1 and the full name prefix of the downloaded data is /service1/movie/movie1/frame, the shared secret can then be used to hide the name components after /service1 and the replied Data payload.

A more systematic data confidentiality solution is to apply name-based access control




Each Data packet carries a digital signature. Sensitive information in Interest name, Data name, and Data content will also be protected.

Figure 4.23: Data-centric security in RapidVFetch

(NAC) [ZYR18]. To be more specific, the service provider acts as an access manager who grants the application the access (i.e., decryption keys) to certain content. The access right granting is based on the service provider's policy and application's identity, which is represented by a public key certificate. By the NAC scheme, the content will be encrypted with encryption keys while the decryption keys will be distributed to authorized applications secretly (i.e., in ciphertext). In addition, different encryption keys protect data under different name prefixes, allowing a fine-grained and flexible access control.

### 4.4.4 Evaluation

We first compare RapidVFetch with existing prefetch approaches. We then simulate RapidVFetch against a baseline scenario where prefetch is not deployed and an optimal scenario where we assume the vehicle will always drive within an RSU's coverage.

**Comparison With Related Works** We made a comparison between existing prefetching approaches with RapidVFetch in Table 4.3.

As shown, unlike existing IP-based and NDN based approaches, RapidVFetch does not assume a known traffic pattern or content popularity distribution and is the first approach that does not require deployments at infrastructure (i.e., RSUs and the wired network). We did not compare our work with others through simulations because many of them utilize traffic/content prediction which highly relies on the environment settings and can be seriously

| System | Main Approach | Cost | T2I* | Prefetch Accuracy |
|---|---|---|---|---|
| [GW16] | Prefetch by Centralized Service | Centralized vehicle info gathering and mobility/content prediction | ✗ | ≈100% when mobility prediction is correct. |
| [GS18a] | Learning Based Prefetch | Information gathering and decision making on each RSU | ✗ | ≈80% when content popularity is known |
| [AB17] | Markov Based Prediction | Centralized Training & Prediction, Cache Redundancy | ✗ | ≈100% when router tree topology is 8 layers high |
| [MG16] | Integer Linear Programming Based | Each RSU prefetches data based on retrieval probability assessment | ✗ | ≈85-90% when content popularity and user mobility are known |
| [RZ13] | RSU Driven Prefetch | Negotiation Between RSUs. Allowing remote control of RSUs. | ✗ | ≈100% |
| Our work | Requester Driven Prefetch | Distributed V2V Communication | ✓ | ≈100% |

∗: Transparency to the Infrastructure

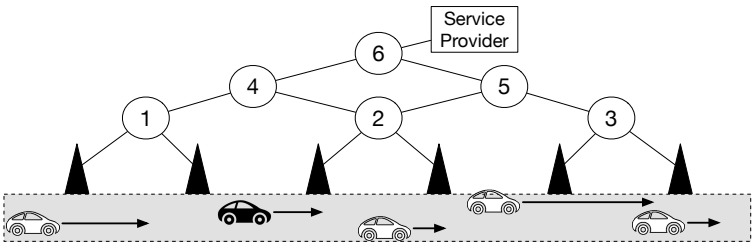
Table 4.3: A comparison between RapidVFetch and related works

affected by biased input, e.g., route prediction on a office worker who drives the same route everyday can be very different from that on a taxi driver whose route is nearly random.

**Simulation Settings.** To evaluate RapidVFetch's performance, we perform simulations over ndnSIM [MAZ17], a NS-3 based simulator. We simulated a 1200 meters single-direction road with 6 RSUs placed at 100, 300, 500, 700, 900, 1100 meter respectively. Each signal-free zone between two adjacent RSUs is 80 meters. The requester starts at 0 meter with a velocity at 20 m/s. To be more specific, we simulated RSUs over IEEE 802.11b 2.4GHz with a signal range of 60 meters and V2V over IEEE 802.11a 2.4GHz with a signal range of 40 meters. In our simulations, the packet loss in wireless media depends on the distance between the transmitter and the receiver. The topology is a simplified tree as shown in Figure 4.24. We have disabled all the NDN cache of non-RSU nodes so that a requester cannot benefit from the cache in common ancestor routers.

In all the simulations, the requester will fetch Data packets one after another, so the performance is not related to Interest sending policies (e.g., batched Interests or parallel

Interests). An Interest packet will be retransmitted in the case of timeout.

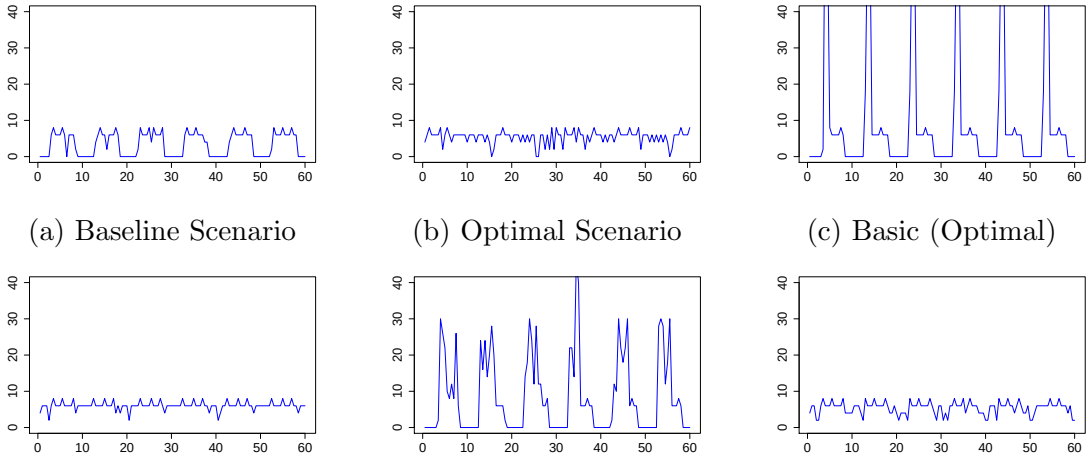



The cache of router 1-6 has been disabled.

Figure 4.24: The simulation topology for RapidVFetch evaluation

**Baseline and Optimal Scenarios.** We first evaluate a baseline scenario (Figure 4.25a) where vehicles download data without RapidVFetch and an optimal scenario (Figure 4.25b) where we temporarily enlarged the RSU coverage to 120 meters so that the requester is always under RSU coverage. As shown, in baseline scenario, the downloading speed is zero when the requester is out of RSU coverage and back to full speed (about 8 pkt/s) after the requester enters RSUs. In the optimal scenario, the downloading speed is always around full speed. Fluctuations in downloading speed are caused by the packet losses in the wireless media.

**RapidVFetch with Optimal Traffic.** We then simulated both basic and real-time RapidVFetch (Figure 4.25c and Figure 4.25d) under an optimal traffic scenario where we temporarily added forwarding vehicles that are always available to the requester. Compared with the baseline scenario, the downloading speed in basic RapidVFetch is greatly improved: after the requester enters a RSU, there is an obvious increase in the downloading speed; this is because the requester can fetch pre-cached packets from the RSU with a minimum delay (fetch within one hop). After that, the downloading speed falls down to a normal downloading speed; this is because all the prefetched packets have been retrieved by the requester and later downloading subjects to the normal RTT. Regarding the real-time RapidVFetch



(a) Baseline Scenario (b) Optimal Scenario (c) Basic (Optimal)

(d) Real-time (Optimal) (e) Basic (Suboptimal) (f) Real-time (Suboptimal)

The x axis is for the simulation time (second) and the y axis shows the downloading speed (packet per second))

Figure 4.25: RapidVFetch performance in baseline and optimal scenarios

in Figure 4.25d, the requester can keep consistently downloading content while moving out of the RSU coverage, so the download speed is close to that of the optimal scenario.

**RapidVFetch with Suboptimal Traffic.** We also simulated basic and real-time RapidVFetch (Figure 4.25e and Figure 4.25f) under a suboptimal traffic scenario when there may not be available forwarding vehicles from time to time. To be more specific, we simulated 9 other vehicles on the road for forwarding packets, each with a random velocity and a random starting position following a Gaussian Distribution model with mean value of 20 m/s (standard deviation is 5 m/s) and 0 meter (standard deviation is 20 meters), respectively. As shown, compared with Figure 4.25c (upper downloading speed exceeds 40 packets/s) and Figure 4.25d, the downloading speeds in Figure 4.25e and Figure 4.25f become lower because when forwarding vehicles are not always available, fewer packets are cached in basic RapidVFetch and fewer packets are forwarded back to the requester in real-time RapidVFetch. Compared with Figure 4.25c, in Figure 4.25e, since a number of packets were not cached scattered among cached packets, the fluctuations in downloading speed are more spread out

and the average downloading speed is lower.

## 4.5 NDN-MPS: NDN Multiparty Trust Support

The majority of the materials contained in this section were modified or quoted verbatim from [].

NDN-MPS is a toolset to support multiparty signature signing and verification over NDN. The core idea is to utilize the names in NDN to express the signing and verification policies required by the multiparty signing scenarios, and use an NDN based remote procedure call (RPC) for signature piece collection from multiple signers. Like a normally signed packet in NDN, a piece of multi-signed data can be verified regardless of where the packet is fetched.

### 4.5.1 Background and Motivation

For reasons such as multiparty contractual policies and mitigating single points of failure, many real-world systems require a joint decision-making process where multiple parties are involved. For example, for grid-connected distributed energy resources (DER) systems, the conventional way of having single proprietary connections to secure grid assets will no longer be sufficient due to the business model change [Joh21, The14]. Instead, smart devices like solar inverters will have multiple parties, including customers, manufacturers, grid operators, who need to access and send remote commands, often over the public Internet.

Recently, Named Data Networking (NDN) [ZAB14] started being explored to provide secure networking support for DER systems. The diversity and number of stakeholders and DER service provider business models [Joh21] require a multiparty trust model with expressive and flexible trust policies. While NDN has developed supporting mechanisms to secure producer-consumer communications by using crypto signature schemes such as RSA and ECDSA, there is no existing work on multiparty authentication and authorization support. This paper fills in the gap by designing and implementing a secure, efficient, and

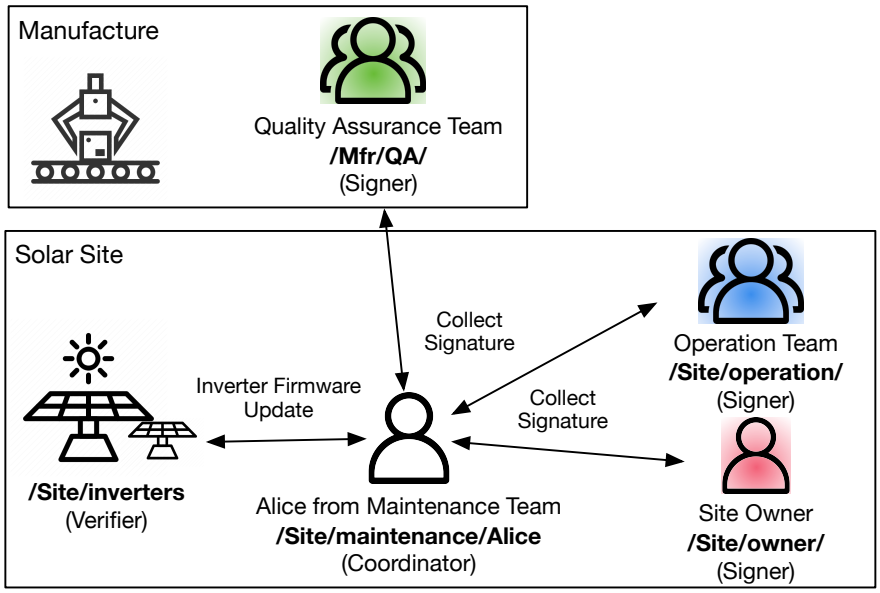



Figure 4.26: An example scenario of multiparty authentication

usable multiparty signature solution for NDN-enabled applications.

We introduce a typical scenario in NDN based DER used throughout the paper to ease the reader's comprehension of various concepts. Assume an equipment manufacturer requests customers upgrade an inverter's firmware to correct a security vulnerability. To make the update, the maintenance operator, Alice, needs a command, an NDN data object, to be jointly signed by the following parties: (i) the equipment manufacturer's Quality Assurance (QA) department (certifying the upgrade is of release quality), (ii) the solar site's operation team (ensuring they know the equipment will be offline for some time), and (iii) the site's owner (permitting the possible production reduction, if the process cannot be completed promptly). The equipment that executes the firmware update will verify the signed data to ensure that it can safely proceed with the update. We further assume Alice's NDN name is /Site/maintenance/Alice, the signers' prefixes are /Mfr/QA, /Site/operation/, and /Site/Owner, respectively. Under each prefix, there can be multiple entities; for example, under /Mfr/QA there can be /Mfr/QA/operatorX and /Mfr/QA/operatorY.

Supporting multiparty signature requires a different security model than that supported

by the existing NDN security solutions [ZYZ18], which considers only two types of parties: a consumer (verifier) authenticates its received data generated by a producer (signer) based on given policies. First, multiparty signing involves third-party signers in the system who need to sign data produced by others. Second, the verifier will need to verify the data's signatures against the given set of signers, so a new type of trust schema is needed to define the set of signers that can jointly make a legitimate signature. Third, effective and secure coordination among multiple signers is required to ensure the correctness of the joint signing process.

Therefore, we propose NDN-MPS, an NDN-based multisignature signing and verification toolset. To simplify the orchestration of the signing process among multiple signers, NDN-MPS makes a design choice of having a coordinator in each multiparty signing process who will take the unsigned data as input and carry out the signing process by collecting required signature pieces from all the required signers. As such, NDN-MPS provides applications with the support of coordinators, signers, and verifiers in multiparty signing scenarios:

1. A new design of trust schema that supports the semantics for multiparty signing and verification, including threshold-based policies (*e.g.*, legitimate if signed by any $k$ out of $n$ signers).

2. A signature collection protocol for the collection of signatures from multiple signers. The protocol ensures the authenticity of the collected signatures and preserves the confidentiality of the data being signed and the secrecy of the signing progress.

3. A new type of key locator scheme to ensure the data integrity in the multiparty signing process and to accommodate multiple signers.

We implement NDN-MPS in C++ with a multisignature scheme called BLS [BLS01, BGW20]. Note that NDN-MPS can easily be extended to other non-interactive multisignature schemes like MSP multisignature [BDN18] and the Bitcoin ECDSA threshold signature [GGK15, GGN16]..

### 4.5.2 A New Trust Model

In each multiparty signing process, there are three parties:

- **Coordinator**. The coordinator is signaled by the data producer to generate a multisignature for the data. Based on the application's requirements defined by the trust schema, the coordinator will collect signatures from a group of known signers and then aggregate them into a multiparty signature. To collect signatures, the coordinator needs to know signers' names and certificates.
- **Signers**. The signers can come from different organizations and have different trust anchors. If a signer approves the data to be signed, either through in-band operations (*e.g.*, database query) or out-of-band operations (*e.g.*, human decisions), that individual will make a signature piece for aggregating with other signers' pieces.
- **Verifier**. The verifier is called by the data consumer to verify a multisignature along with the data covered by the signature. The verification succeeds if the multisignature is valid and if its signers satisfy the application's requirements as defined in the trust schema.

It is noteworthy that we separate the above roles based on functional logic. In practice, one entity can play multiple roles, *e.g.*, the coordinator can also be one of the signers. In addition, an entity can play different roles in different signing processes.

In our example, the operator Alice is the producer of the command and runs the coordinator. The manufacturer's QA team, the site's operation team, and the site owner are the signers. After being signaled by Alice that the signed command is ready, the inverter, as the data consumer, will fetch the data and call the verifier to verify the multisignature.

**Assumptions** We assume all three types of entities have gone through the standard NDN bootstrapping process [ZYZ18], through which each entity in the system obtains an identity, certificate, the system trust anchors, and the trust schema. Note that all the signers' key

types should be compatible with NDN-MPS, *e.g.*, BLS key type with the current NDN-MPS implementation. Therefore, signers can authenticate coordinators, and verifiers can authenticate signers. As stated, we also assume the coordinator's awareness of the signers from which the coordinator can select appropriate signers to generate legitimate signatures. In addition, there is a known attack against the BLS signature, called rogue key attack [BGW20], where a malicious coordinator may successfully generate a legitimate multisignature without obtaining real signature pieces from the signers. To prevent such types of attacks, as suggested by [BGW20], we assume that each signer's certificate carries a piece of cryptographic information, called a proof of possession, to prove the possession for the public key corresponding to secret key.

**Signing and Verification Process** Under the system model, a complete signing and verification process can be described as follows in an abstract way.

1. The coordinator starts by deciding the signer list based on the trust schema and proceeds through the signing process by contacting individual signers.
2. Each signer verifies the request against the trust schema to ensure that the coordinator is legitimate.
3. If the signer is permitted by the application logic to sign the data, it generates the signature and sends it to the coordinator.
4. The coordinator aggregates the collected signature pieces into one and appends it to the unsigned data to get the finalized signed data object.
5. After the consumer application obtains the signed data, the verifier verifies the signed data against the trust schema.

The coordinator is triggered to start the process by certain application events. For example, the maintenance operator Alice can start the coordinator when receiving a firmware

update notification from the manufacturer. Similarly, obtaining the signed packet is also realized at the application layer. In the same example, the inverter can call the verifier when receiving the firmware update command.

To ensure the security of the system, the following policies should be explicitly and rigorously defined by the trust schema.

- **Coordinator Verification Trust Schema**. In step 2, the trust schema needs to define the coordinator based on the names or name patterns of legitimate coordinators.
- **Multisignature Trust Schema**. In step 5, the trust schema needs to specify how many and which signers can generate a legitimate signature for a given named data.

In step 3, whether a signer will agree to the sign data depends on two factors. First, each signer should ensure the data is allowed to be signed by her according to the installed multisignature trust schema. Second, the signer may also need approval from the application logic, *e.g.*, through some database query or even human decision making; this process is application-specific and thus omitted in this paper. Importantly, depending on application scenarios, the time duration of this step may vary significantly. In the example, to ensure the power supply after shutting down the inverters for the update, the public utility may need to measure the remaining power supplies against the power demand in the area before agreeing to sign the command.

### 4.5.3 Design of NDN-MPS

NDN-MPS is designed as a security support toolset for applications. NDN-MPS consists of three main components (Figure 4.27). First, a new type of trust schema that allows applications to express the signing and verification policies of multisignatures. Second, an extension of the key locator in NDN data objects to keep the to-be-signed data consistent across signers and allow complex signature information. Third, a signature collection protocol for the coordinator to collect signatures from multiple signers.

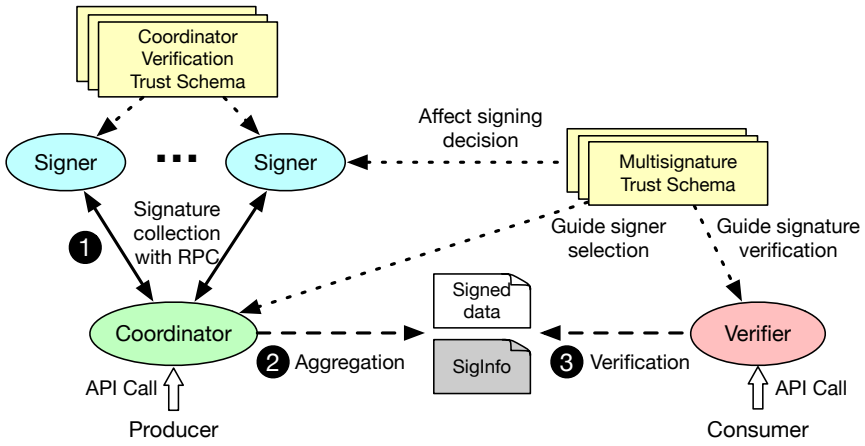



Figure 4.27: An overview of NDN-MPS

These three components are used in the process of multisignature signing and verification.

**Signature Collection** The coordinator starts the signing process when the application provides the unsigned data and the multisignature trust schema through a local API call. From the trust schema, the coordinator identifies a list of signers and then starts the signature collection. In a nutshell, the coordinator issues a signed request to the signer and presents the unsigned data as the request parameter. To ensure the consistency of the data to be signed among signers, the coordinator will fill the key locator field with a placeholder name called late-binding key locator. Such a placeholder is the key to make the data to be signed deterministic and consistent across multiple signers.

Upon receiving the request, each signer will first verify it against the coordinator verification trust schema to ensure that its sender is a legitimate coordinator. After that, the signer can decide on whether to sign the data or not. In this process, besides checking against known multisignature trust schema policies, the signer may also need to go through some application-specific procedures. If the signer's application agrees to sign the data, it will generate the signature over the data, without modifying the key locator or any other fields.

In an unstable network or in an attack, signers may be forced offline, unable to be reached by the coordinator. In this case, the coordinator can try to reach an alternative

signer permitted by the trust schema, *e.g.*, the coordinator Alice can try /Mfr/QA/operatorY when /Mfr/QA/operatorX is unavailable. This design provides a certain degree of resiliency that even if some signer fails, the signing can continue. Importantly, the change of signers will not affect the value of the late-binding key locator name as it is a placeholder, so the consistency of the data being signed among signers is preserved.

**Signature Aggregation** After collecting sufficient signature pieces, the coordinator aggregates them to generate a single signature and attach it to the original unsigned data. In addition, the coordinator generates another Data packet, called a SigInfo packet, to carry carries the signer information, *i.e.*, signers' key names. The referenced trust schema policy name and the aggregated public key value can also be included to facilitate the verification process. Importantly, SigInfo is named with the placeholder key locator name. As such, a SigInfo packet can be linked from the signed data's key locator, overcoming the limitation of the existing key locator design, which does not support multiple signers.

**Signature Verification** To verify a piece of multiparty signed data, the verifier first needs to obtain both the signed data, which is expected to be integrated with specific applications. The SigInfo packet can then be fetched with an Interest carrying the key locator name specified in the signed data. After that, the verifier first checks the signer list from the SigInfo packet against the multisignature trust schema to ensure the signers are legitimate to sign such a piece of data. Finally, the verifier validates the multisignature using the public keys of the signers.

### 4.5.4 Multisignature Trust Schema

As the baseline, the trust schema allows specifying multiple required signers. For example, a signature is valid only when required signers $S_1$ to $S_n$ are present. With our trust schema syntax, the key name of $S_1$ to $S_n$ will be listed under a configuration field called "all-of".

Furthermore, to enjoy the flexibility of structural names, the trust schema should allow name patterns. For this purpose, the multiparty trust schema supports wildcard character $*$, which can match any name component. In our example, the wildcard character in /Mfr/QA/*/KEY/* allows any operator under the equipment manufacturer's QA team to participate in the joint signing.

The pseudo-code of a multisignature trust schema for the example use case is shown below. The rule applies to the command to update the solar inverter firmware. The known signers should be the registered operators under each responsible organization's prefixes, which are omitted for simplicity.

```
Data profile: /Site/inverters/firmware/update
All-of {    /Mfr/QA*/KEY/*
            /Site/operation/*/KEY/*
            /Site/Owner/*/KEY/*    }
```

we let multisignature trust schema support threshold verification policy by adding two new field called "at-least-num" and "from". To express the policy that a signature is valid when $k$ out of $m$ signers are present, the $k$ will be specified by the "at-least-num" and the key names of $m$ signers are listed after the "from". We show an example of multisignature trust schema with threshold verification below. Different from the required signers, it only needs any two of the three parties to generate a legitimate signature. Note that these two fields can also be used together with existing multisignature fields to support more complex policy, *e.g.*, a signature needs $n$ requires signers and $k$ out of $m$ optional signers.

```
Data profile: /example/data/*/*
At-least-num 2
From {     /Site/operation/*/KEY/*
           /Mfr/QA/*/KEY/*
           /Site/Owner/*/KEY/*   }
```

Compared with cryptographic solutions of threshold scheme, NDN-MPS does not need any centralized trusted dealer, and the key maintenance is much simpler. To be more specific, the key setup does not require interactive operations and can be done individually per signer with the public parameters. In addition, the keys can be managed individually. For example, a signer's key can be renewed or revoked without bothering other signers' keys.

### 4.5.5 Evaluation

**Implementation** We have implemented NDN-MPS into an open-source library in C++. Our library depends on ndn-cxx [Ndn21] and a BLS library [Shi21] that have been fully examined and is compatible with the BLS API specification defined by Ethereum 2.0 [Pro21]. Our library is implemented over the curve BLS12-381. In this section, we explain our implementation of multiparty trust schema and the integration of NDN-MPS into NDNCERT to realize multiparty-approved certificate issuance and revocation.

The multisignature trust schema is implemented in the library as a trust schema configuration file parser. To support more complicated use cases where there are multiple acceptable signer sets, one can simply created multiple trust schema files applying to the same data prefix. In the signature verification process, the signature is accepted if there is at least one trust schema rule is satisfied. The known signers are implemented with a separate configuration file which contains a list of NDN certificates.

**Bandwidth and Network Overhead** A BLS signature piece from a single signer and an aggregate signature are of the same size. With BLS12-381 elliptic curve [SKS20] (providing about 128 bits of security), the BLS signature size is 96 bytes, which is slightly larger than a ECDSA signature (about 72 bytes) and much smaller than a RSA signature (256 bytes) when providing similar bits of security. Regarding the size of public key certificates, a BLS public key over BLS12-381 is only 48 bytes, which is much smaller than either ECDSA public key or a RSA public key when providing similar bits of security.

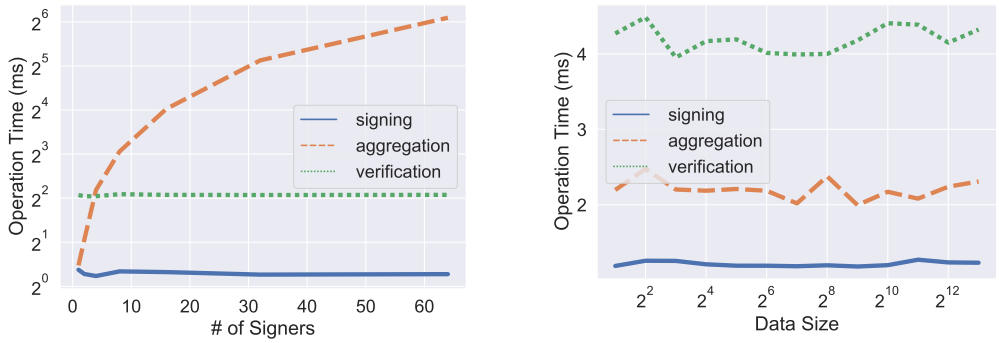
The RPC in NDN-MPS requires 2 round trip time (RTTs) when the signer is available. If the signer is unavailable, the coordinator will notice it after the timeout of the first Interest.

| Operation | BLS | ECDSA | RSA |
|---|---|---|---|
| Signing | 1.38ms | 0.15ms | 1.58ms |
| Verification | 4.32ms | 0.43ms | 0.14ms |

Table 4.4: The overhead comparison between BLS in NDN-MPS and conventional signature schemes in NDN-CXX

**NDN data Signing/Verification Overhead** We compare the NDN data signing and verification performance of BLS multisignature with conventional public key signatures including ECDSA (on elliptic curve secp256r1) and RSA (with 2048 bits key). ECDSA and RSA are realized by NDN-CXX's keychain with OpenSSL. The content of the data to be signed is 16 bytes of random data.

As shown in Table 4.4, although slower than ECDSA and RSA (OpenSSL has specialized hardware and software optimization), the performance of BLS signature in NDN is still practical.




(a) Different # of Signers (b) Different data Size

Figure 4.28: NDN-MPS scalability to signer number and data size

**Scalability to Signer Number and Data Size** We also evaluate the scalability of NDN-MPS by increasing the number of signers needed for the signature generation and the size of data to be signed. As shown in Figure 4.28a, only aggregation time is linearly affected by the signer number while the signing and verification time stays almost constant. As reported in Figure 4.28b, with two signers, when increasing the data size, time of signing, aggregation and verification stays mostly constant.

# CHAPTER 5

# Build Usable Security Solutions using Data-centric Security

In this section, we describe the design of EL PASSO and ReLiShare, and discuss how the concept of data-centric security can be utilized to address security and privacy issues encountered by today's application systems.

## 5.1 EL PASSO: Privacy-preserving Asynchronous Authentication

The majority of the materials contained in this section were modified or quoted verbatim from [ZKS21].

EL PASSO is a privacy-preserving and asynchronous Single Sign-On (SSO) protocol. The insight of EL PASSO is to decouple the SSO based user authentication from the synchronous secured connection to the identity provider by making the authentication information a piece of secured data. This secured data is considered as a credential and will be carried with the user for each sign on process.

### 5.1.1 Background and Motivation

SSO is an answer to the complexity and fragility of using individual passwords on the web, *i.e.*, leading to reuse and leaks [IWS04]. SSO enables the use of a unique identity provided by an Identity Provider (IdP). Users authenticate themselves to services (called Relying Parties–

RP) with tokens provided by their IdP. SSO improves overall web security [Goo12] and enables the generalization of good security practices such as the use of 2-factor authentication (2FA) [OBM18].

OpenID Connect (OIDC) is a dominant SSO solution used by over a million websites in 2020 [Sim20]. Major web players such as Google or Facebook play the role of IdPs, offering so-called *social login* features to RPs previously registered with their services. However, while facilitating identity management, the wide adoption of OIDC raises concerns on users' privacy [Bra07, GN14]. These concerns are direct consequences of the synchronous and coupled mode of operation of OIDC, illustrated in Figure 5.1. Each login request to an RP requires first an interaction between the user and the IdP for authentication and then another interaction between the RP and the IdP to validate credentials. An IdP is, therefore, aware of its users' every sign-on attempt and can infer private information from the nature of visited websites [KMW07]. Similarly, RPs learn users' identifiers at IdPs systematically, enabling to aggregate identity elements shared to different RPs and violate *purpose limitation* (*e.g.*, as defined in the EU GDPR [Com16]). OIDC extensions have been proposed to increase users' privacy but only partially address these problems, as they either still leak users' global identifiers to the RPs [FKS15] or do not prevent the IdP from logging visited

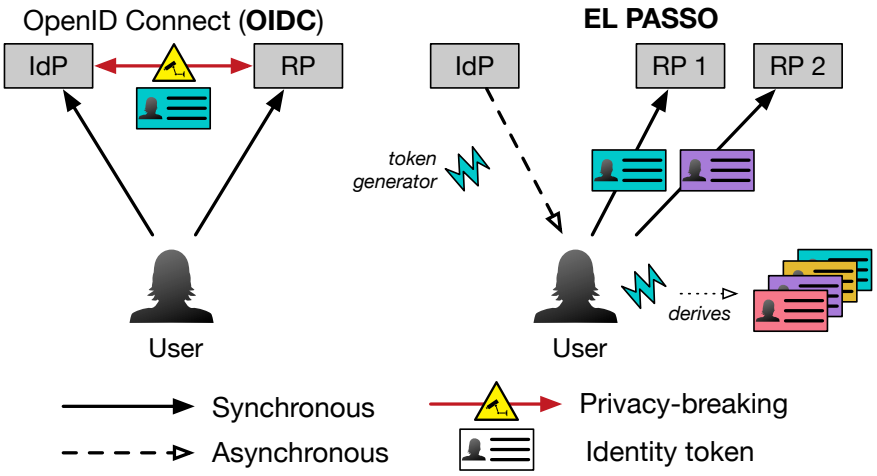



Figure 5.1: Sign-on in OIDC and in EL PASSO.

websites [App20]. We note, finally, that in addition to privacy concerns, the synchronicity in OIDC impacts *availability*: Users simply cannot connect to an RP if their IdP is offline. This requirement of high availability can prevent small organizations (*e.g.*, digital rights NGOs) from offering an alternative to tech giants' IdPs and counter Internet consolidation [ATN19].

Our objective in this paper is to bridge the gap between the simplicity, ease of deployment and practicality of OIDC, and privacy-preserving SSO based on anonymous credentials. We target a complete and integrated system for privacy-preserving SSO that does not compromise performance or ease of use in favor of security.

### 5.1.2 Design of EL PASSO

We present the design, security analysis, and evaluation of EL PASSO, a practical privacy-preserving SSO system. EL PASSO is asynchronous and offers unlinkable authentication and the strong privacy guarantees of anonymous credentials. The generation of authentication material by the IdP is decoupled *in time* from its use by the client to sign on at some RP. It enables minimal disclosure of information: Users may share only elements necessary for a specific RP or provide authenticatable personal properties to an RP, such as being above a minimum age or coming from a certain geographical area, without sharing their exact age or location. The design of EL PASSO acknowledges the practical consideration that *unbreakable* anonymity is not desirable for many online services. EL PASSO offers guardrails to the risk of digital impunity associated with minimal disclosure of information, by providing optional support for *accountability* guarantees to RPs about users signing on their services. Users convicted of fraudulent behaviors (*e.g.*, authors of hate speech or harassment in an online forum, or publishers of illegal content) can be eventually identified. This identification obeys a strict cooperation process involving *several* authorities, whose number and identity must be announced *a priori* by RPs using the feature.

EL PASSO provides support for features that are typically expected from non-privacy-preserving SSO. We develop a double-secret scheme to enable IdP-backed, multi-device de-

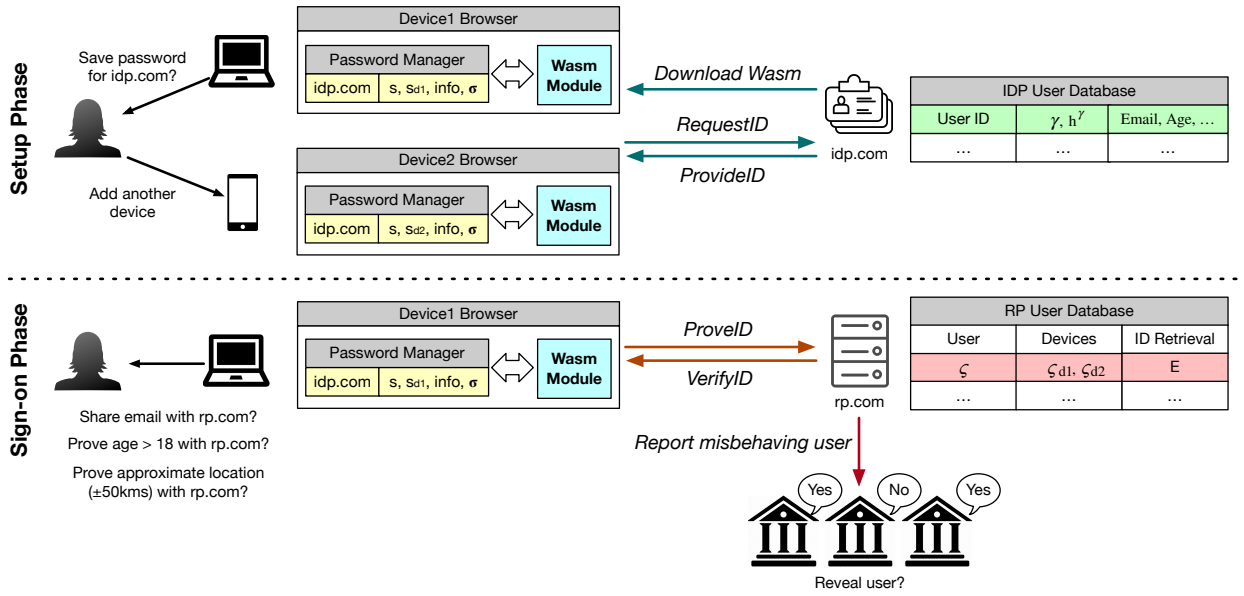



Figure 5.2: An overview of EL PASSO.

ployments. EL PASSO is robust against the theft or loss of a device and the secrets it contains and exposes simple recovery procedures at the IdP. The usage of device-specific secrets naturally supports 2FA without disclosing the user's phone or email address. In addition, we provide guidelines for incremental deployment and show how EL PASSO can operate alongside OIDC.

EL PASSO operations are divided into two asynchronous phases (Figure 5.2). In the *setup phase*, the client obtains an anonymous credential from the user's IdP. In the *sign-on phase*, the client prepares an RP-specific derivation of this credential based on what information the user decides to disclose, and proves the authenticity of this client to the RP. The setup phase is executed periodically (*e.g.*, once every few days), while the sign-on phase is executed each time the user logs in or creates an account on an RP.

All user-side operations (*i.e.*, cryptographic operations) are automatically handled by a WebAssembly (Wasm) module. Wasm is an open standard that defines a portable binary-code format for executable programs that can run natively in all major web browsers. The module is referenced in the HTML pages provided by RPs and IdPs and automatically

downloaded, run and later cached by the browser without user interaction. If one of the participants does not support EL PASSO, our system falls back to regular OIDC interactions.

**Setup Phase** The user first generates its secret $s$ used later for authentication. The user authenticates to their IdP and request their credentials bound to $s$. The credentials include user attributes (i.e., name, age) verified by the IdP and a long-term pseudonym $\gamma$. Importantly, the credentials can be used only when possessing the user secret $s$, so an IdP (or anyone else) is unable to connect to RPs on behalf of the user. Both the secret $s$ and the obtained credentials are automatically serialized by the Wasm module and stored in the browser built-in password manager under the DNS domain of the IdP. The user does not have to manage cryptographic material manually.

**Sign-on Phase** The user connects to the RP's login page that lists the information requested for login (under the principle of purpose limitation [Com16], requested information should be only what is strictly necessary to provide the service). The Wasm module intercepts the request and asks the user for their consent to share the requested information, using a popup.

If approved, (i) the module locally randomizes the credentials so that even if an attacker observes both the RP and the IdP it cannot link the credentials to a specific user of this IdP. (ii) The module then provides the RP with the randomized credentials and selectively discloses any subset of information embedded in the credential, enabling *selective attribute disclosure*. The client may also generate and include proofs about *properties* of the attributes they do not wish to share in the clear, enabling *provable personal properties*. The secret $s$ is never shown to the RP; instead, the client proves knowledge of it through a zero-knowledge proof (which unconditionally hides $s$). Note that the client cannot generate fake attributes without forging credentials, which would require breaking the underlying PS signature scheme. (iii) The client locally generates a RP-specific pseudonym $\zeta$. It is impossible

to generate multiple pseudonyms for the same RP DNS domain providing *intra-domain linkability*. Once the RP verifies the credential along with the pseudonym, it considers the user as authenticated. The credentials and the user secret required for the sign-on are automatically provided by the browser password manager.

Finally, if support for *reliable identity retrieval* is required by the RP, the client provides and proves the correctness of an El-Gamal encryption $E$ of their long-term pseudonym $\gamma$ encrypted under the public key of specific decryption authorities. If the user misbehaves, the RP discloses $E$ to these decryption authorities, which decrypt it and collaborate with the IdP to recover the identity of the user. EL PASSO supports flexible key management for decryption authorities—the ciphertext is typically encrypted using *threshold encryption*, where at least a threshold number of decryption authorities are needed to recover the user's identifier.

**Multi-device Support** A user may add a new device and use it to connect to RP accounts created with any of their older devices. Users first connect to their IdP with the new device using the standard login procedure (*e.g.*, username/password) so that the IdP can associate the new device with the old ones. Then, the new device needs to receive the secret $s$ without leaking it to any third party; this is achieved as follows. The new device generates an ephemeral public/private key pair and sends the public key to the IdP. The user confirms the new device at the IdP using one of the older devices, and the Wasm module encrypts $s$ under the new device's public key. The IdP sends the encrypted secret to the new device allowing it to request credentials over $s$. The process does not require direct communication between user devices and can be performed asynchronously. The users confirms the new device with a single button at the IdP without being exposed to the key exchange.

**Two-factor Authentication** EL PASSO allows RPs to require two-factor authentication (2FA), *i.e.*, that users connect from two different devices for attesting their authenticity.

Under the principle of minimal disclosure of information, 2FA does not require revealing an email address or phone number, but only to use two different previously-enabled devices. This requires, in addition to secret $s$, a device-specific secret $s_d$. The client includes $s_d$ during the setup phase making it a part of the credentials. When the user connects with a given device to the RP for the first time, they provide $\zeta$ and generate an RP- and device-specific pseudonym $\zeta_d$ derived from $s_d$ and the RP's DNS domain name. Similarly to $\zeta$, $\zeta_d$ are unlinkable across domains and cannot be re-used by a malicious RP. The RP is able to link the new device to the user account using $\zeta$ and adds $\zeta_d$ to the list of authorized devices. Subsequent logins using the same device requires only providing $\zeta_d$ and do not involve additional overhead. When requiring 2FA, an RP simply checks that two subsequent logins are performed from devices with different values of $s_d$.

**Device Theft Recovery** A user can declare the loss of a device to their IdP. The IdP will stop issuing credentials for that device. A thief able to unlock the secret storage of the stolen device's browsers would be able to connect to RPs, unless 2FA is required, but only until the IdP credential expires. It will not be able to authorize new devices. Users do not lose access to their RP accounts as long as they hold at least one device (or two devices, if 2FA is required). A user can replace their secret $s$ using the following procedure. The user contacts the IdP and asks for credentials on a new, blinded $s'$; from now on, the IdP will not renew credentials for $s$ to preserve sybil resistance. The client connects to the RP and presents credentials over the old, expired $s$, $\zeta(s)$, and credentials over the new $s'$ and $\zeta(s')$. The RP replaces $\zeta(s)$ by $\zeta(s')$, and stops accepting credentials on $s$. The user does not need to replace the secret at all the RPs immediately, as the attacker is not able to use expired credentials.

### 5.1.3 Evaluation

EL PASSO aims for ease of deployment by RPs and IdPs, and simplicity for its users. User-side operations are supported by a client module in WebAssembly [HRS17] received on the fly from the IdP and cached in the user's browser. The module automatically manages cryptographic material, storing it using the browser built-in password manager. As a result, our platform does not require prior software installation or specific hardware and exposes user interactions similar to OIDC.

EL PASSO is built using PS signatures [PS16, PS18] and designed to limit the amount of heavy cryptographic operations required for all parties. Our evaluation using representative user devices and RP and IdP services hosted on Amazon EC2 indicates that EL PASSO performance, costs, and scalability make it amenable for large-scale deployments. Sign-on operations only require one round-trip between the user-side client and the RP, and while more computations are required at the user side than for OIDC, their CPU cost is a factor of 39x to 180x lower than for those of IRMA [BBC09, ABH17], a previous scheme using anonymous credentials. This results in comparable or even lower sign-on latency compared to OIDC, *e.g.*, only 250 ms on a laptop and 800 ms on a Raspberry Pi representative of a mobile device. Finally, implementations of the RP and of the IdP scale vertically and horizontally in the cloud, and allow throughput of more than 260 setup phases or more than 170 sign-on phases per second using only a 4-core VM.

More evaluation results can be found in [ZKS21].

## 5.2 ReLiShare: Data Sharing with Reliable Leaker Identification

The majority of the materials contained in this section were modified or quoted verbatim from [ZXG20].

ReLiShare is a data sharing systems to allow accurate leaker identification if one or more

participants in the sharing leak the data in the future. Our insight of ReLiShare is to let the dataset itself carries security properties that can benefit a later leaker identification process in case of a leakage event.

### 5.2.1 Background and Motivation

As the world becomes increasingly online, a wide spectrum of applications need to share datasets containing sensitive information (e.g., Personally Identifiable Information (PII) [MGS10b]) with multiple parties. For example, public Medicare and Medicaid Services (CMS) share data that contains people's identifiable information with laboratories for research purposes across the country [Cen20]. More than 60% of hospitals in the US share patient information with different health care providers [Bla19] to benefit patients, e.g., avoiding duplicate tests. It is also increasingly common that schools or employers share student/employee information with online educational services for specialized training.

While most data sharing systems entrust the authorized parties with keeping data confidential, recent years have seen increasing violations of this trust [The19b, The19a, Doo20a, BBC19, Doo20b]. For example, in 2019, a data breach affecting approximately 4.9 million users was caused by the leakage from third party services [Doo20a]. As required by new regulations and laws such as GDPR [EU 19] and CCPA [Cal19], whenever a data leakage occurs, the leaker(s) must be reliably identified.

The leaker identification (also called traitor identification or guilty agent detection) problem can be defined as follows. A data owner (called a sender) sharing a set of data objects with data users (called receivers), where the parties on both sides agree that the shared data shall not be disclosed to any third party. If any, or all, of the shared data objects are found in an unauthorized place (*e.g.* the public Internet, personal laptop), the leaking source(s) should be identified.

A number of solutions [AK02, AHK03, LSJ03, LSJ05, LWD04, GWL06, Tar08, XF09,

PG09, PG11, GS18b, GS19, SG20] have been proposed to identify leakers in dataset sharing, however several critical challenges remain unsolved.

1. First, most of them assume a *trusted data owner* who would never leak data. The solutions with this assumption cannot detect leakage from the data owner, for example, by its employees intentionally or unintentionally [The19b, The19a, Doo20b] or due to vulnerable firewalls [BBC19]. Worse yet, since the data owner knows the exact dataset shared with each receiver, it could frame a data receiver by intentionally leaking a dataset that is unique to an innocent receiver.

2. Second, most existing approaches cannot handle leakage by a set of *colluding* parties. Through collusion, multiple leakers can figure out the distinction in datasets by comparing each one's received data, then erase the distinction, for example, by only leaking common data objects [HPC10].

3. Third, non-repudiation of multiparty data sharing remains a challenge. In the absence of immutable evidence showing which receiver possesses which dataset, a leaker may deny its possession of certain data pieces.

In this paper, we propose ReLiShare, a data sharing system that overcome the above shortcomings in data leaker identification. Compared with the existing results, ReLiShare provides non-repudiation of data possessions of every party in a data sharing agreement; and given a data leakage, ReLiShare can reliably identify the responsible party, or a responsible set of colluding parties, even in the case of the data owner being the guilty party or among them. More specifically, the ReLiShare design includes the following three basic approaches.

- *Oblivious Data Sharing.* The sender shares data objects through Oblivious Transfer (OT) [Rab05] with each receiver, so that the sender does not know the exact set of data objects is transferred, and the receiver does not know the the complete set of data objects sent by the sender.

- *Non-repudiable Receipt based on Merkle Tree.* During the data sharing process, ReLiShare immutably records the result of the data sharing with Merkle Trees and generates a digital credential, preventing any party from lying or denying what data objects have been transferred/received. This step involves the use of an honest third party, called a notary.
- *A Knowledge based Identification Algorithm.* We propose a novel identification algorithm to identify leakers by inferring the knowledge of the leaker(s) from the leaked data objects using a statistical method called the binomial test. With this identification algorithm, ReLiShare can effectively pinpoint guilty parties, including (i) a sender, (ii) non-collusive receivers, or (iii) collusive receivers, with a false positive rate and/or a false negative rate.

### 5.2.2 Design of ReLiShare

The high-level idea of ReLiShare is to identify leakers based on system participants' different knowledge of the dataset: In the data sharing process, ReLiShare will let every party, including the sender, have a different view of the dataset. Importantly, each party's knowledge is unknown to others unless there is collusion. In addition, ReLiShare will immutably record the data sharing process to ensure non-repudiation of the identification results because leakers cannot deny or lie about their knowledge of the dataset.

To meet the target properties of ReLiShare, we propose three main technical approaches (Figure 5.3).

The first approach is oblivious data sharing. For each receiver, the sender shares a randomly-permuted dataset to the receiver with 1-out-of-2 OT and the receiver will randomly generate the choices used in OT (❶). By properly preparing the OT messages, each receiver will receive a distinct dataset unknown to the sender and other receivers.

The second approach is indisputable receipt based on Merkle Tree of data sharing. Each data sharing process will be proved immutably by a Merkle-Tree based credential generated by a Notary (❷). Such a credential logs the data objects that have been sent and received,

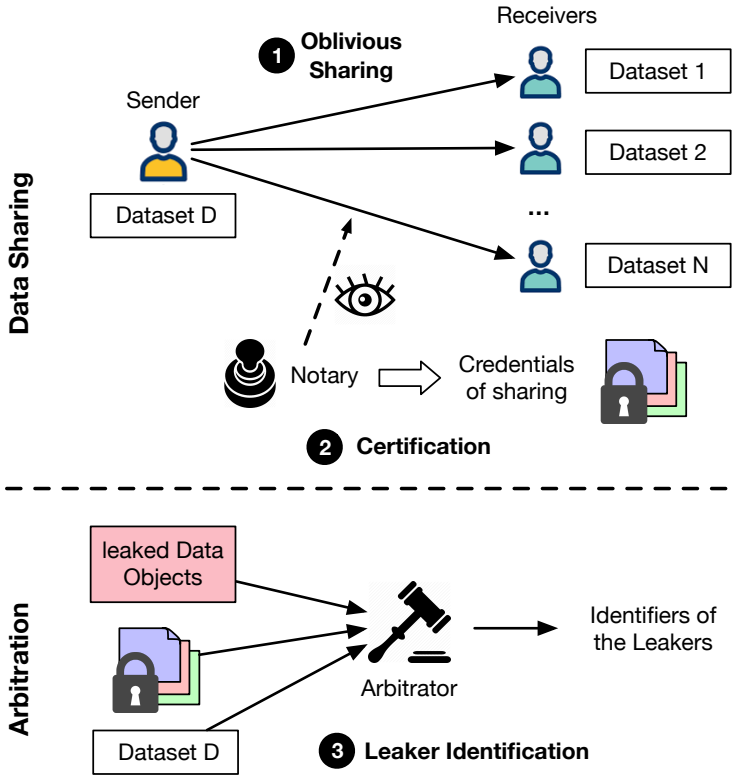



Figure 5.3: An overview of ReLiShare

preventing any party from denying or lying about the data allocation. In this process, the Notary cannot access the plaintext dataset because all data objects will be hashed into the Merkle Tree.

The third approach is knowledge-based keaker identification. In the event of a data leakage, by reversely inferring leakers' knowledge from the leaked data using binomial test, ReLiShare can identify all leakers with high accuracy (❸).

After the sharing process, by employing OT, each received copy is a different collection from all others; even a combination of several received datasets is also distinguishable from other combinations. Importantly, no single party is fully aware of the allocation among the receivers. In addition, with credentials published by the Notary, no party is able to cheat in a later arbitration. Due to these properties, in the case of a data leakage, the knowledge-based identification algorithm is able to effectively and indisputably identify all leaking sources.

To differentiate the knowledge of a dataset among the sender and all receivers, in ReLiShare, some data objects are not commonly shared, namely dropped during the sharing process. This is intended and usually inevitable because the goal is to add difference to each received dataset by doing data object distribution [PG09, PG11], which can potentially limit the use cases. To allow a wider adoption, ReLiShare provides two options for the sender: the sender can (i) sacrifice the completeness of real data objects, i.e., each receiver only receives a subset of the genuine dataset, or (ii) add synthetic data objects into the dataset so that each receiver can receive all genuine data.

### 5.2.3 Evaluation

We implement a prototype of ReLiShare sharing system in C++ and the knowledge-based leaker identification algorithm in Python. Our experiments using real world datasets show the efficiency and effectiveness of ReLiShare. With 1-out-of-2 OT extension [ALS13] and the Handyman dataset [Hea19], it only takes about half a minute for a sender to share more than one million data objects to a receiver. The computation and network overhead for generating Merkle Tree based credentials are also practical to be deployed. The identification evaluation results show that ReLiShare can provide $> 99.99\%$ identification accuracy in different leakage scenarios with a relatively small number of data for allocation.

More evaluation results can be found in [ZXG20].

## 5.3 Moving Towards Data-centric Security

Duo to the constraints of today's Internet architecture deployment, EL PASSO and ReLiShare are implemented over TCP/IP and thus is a mix of two security models. For example, in EL PASSO, the client needs to set up a secured channel to the service website because this is required by the modern web browsers (*e.g.*, Chrome marks all non-HTTPS websites as unsafe). Such a mix can be found in other data-centric security solutions in today's Internet,

like the signed HTTP objects in the Web Package, the data objects in IPFS, the verifiable credential object in Verifiable Credential. The discrepancy between the two security models will inevitably result in more complexity. For example, when fetching a signed HTTP object, the client still needs to decide which IP address to connect. Similarly, in IPFS, a mapping between an IPFS content identifier and IP address is also needed to fulfill the mismatch between channel and data centricity.

In comparison, NDN's way of using named secured data and stateful forwarding is to remove the discrepancy by providing a coherent data-centric network-to-application solution suite.

# CHAPTER 6

# Conclusion

In this dissertation, we describe the two security models and analyze their difference from the application perspective. We highlight that what applications need is secured data rather than secured channels, and it is an observed new trend that both the network community and application community are moving towards the same direction from channel-based security to data-centric security.

When implementing data-centric security in the application layer over today's Internet, it can be a signed HTTP object, an EL PASSO credential, and so on. However, additional complexity is needed for application-layer data-centric security solutions to fit today's deployed Internet architecture and the security model. The root cause is the discrepancy between the deployed channel-based network and network security model and the application layer data-centric security solutions. In comparison, NDN implements data-centric security in both the network and application layer in a coherent way, by making named secured data the fundamental building block of communications and adopting stateful forwarding. NDN provides a data-centric communication model by naming the secured data for networking purposes, thus removing the discrepancy.

Through the systems we described in this paper, we show the new way of doing network security, namely, using NDN's named secured data and stateful forwarding, can be used to address security issues that are intractable for channel-based security. Specifically, we confirm the desired properties of NDN security:

- Names provide rich application layer semantics and allow to construct fine-grained

security policies

- Secured data is friendly to asynchronous communication.
- Stateful forwarding provides traffic state and insights that are needed for DDoS mitigation.

Regarding the deployment of NDN and NDN security, there are two feasible ways. The first way is to start deployment from edge networks, including smart home networks, vehicular networks, data center networks, etc. This is because NDN's named secured data allows simple and efficient security solutions in edge networks, providing incentives for stakeholders. In addition, the deployment at the edge does not dependent on the NDN deployment in the Internet. An example is the Sovereign, which can facilitate the security and privacy of smart home systems, and the deployment cost is low as it requires no infrastructure change. When scattered end systems start to adopt NDN based systems, it can push a larger and deeper NDN deployment to connects the NDN islands.

The second way is to build overlay NDN networks on the current Internet. The first potential motivation is DDoS mitigation. As shown in Section 4.2, FITT can be deployed over today's CDN infrastructure without hardware update, and at the same time, offer incentives to CDN service providers by supporting both CDN and DDoS functions with a lower resource overhead. Other motivations of overlay NDN networks can be distributed applications. Well-known overlay networks that have been successfully deployed include (i) the overlay P2P networks deployed with BitCoin, Ethereum, and other cryptocurrency systems, and (ii) the P2P network used for onion routing. The commonality of these overlay network systems is they are needed by a certain distributed application (*i.e.*, cryptocurrency and onion routing). Therefore, to push a large-scale NDN overlay deployment, an important aspect is to explore the NDN based distributed applications. The design of DLedger has shown the potential benefits of using NDN as the underlying P2P network for a distributed system.

Future works can continue to address more security issues with the new way of providing security, namely the data-centric security model and especially NDN's named secured data. Considering that the data-centric security model fits non-infrastructure networks, NDN security researchers can focus on building usable security solutions to edge networks. In addition, since the use of secured data alleviates the dependency of centralized services, another aspect of the future work can be to support distributed application systems.

# REFERENCES


[AAB17] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. "Understanding the mirai botnet." In *USENIX Security Symposium*, 2017.

[AB17] Noor Abani, Torsten Braun, et al. "Proactive caching with mobility prediction under uncertainty in information-centric networks." In *Proceedings of the 4th ACM Conference on Information-Centric Networking*, pp. 88–97. ACM, 2017.

[ABB18] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. "Hyperledger fabric: a distributed operating system for permissioned blockchains." In *Proceedings of the thirteenth EuroSys conference*, pp. 1–15, 2018.

[ABC19] Josh Aas, Richard Barnes, Benton Case, Zakir Durumeric, Peter Eckersley, Alan Flores-López, J Alex Halderman, Jacob Hoffman-Andrews, James Kasten, Eric Rescorla, et al. "Let's Encrypt: An Automated Certificate Authority to Encrypt the Entire Web." In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pp. 2473–2487, 2019.

[ABH17] Gergely Alpár, Fabian van den Broek, Brinda Hampiholi, Bart Jacobs, Wouter Lueks, and Sietse Ringers. "IRMA: practical, decentralized and privacy-friendly identity management using smartphones." In *10th Workshop on Hot Topics in Privacy Enhancing Technologies*, HotPETs, 2017.

[AHK03] Rakesh Agrawal, Peter J Haas, and Jerry Kiernan. "Watermarking relational data: framework, algorithms and analysis." *The International Journal on Very Large Data Bases*, **12**(2):157–169, 2003.

[AK02] Rakesh Agrawal and Jerry Kiernan. "Watermarking relational databases." In *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases*, pp. 155–166. Elsevier, 2002.

[Aka19] Akamai. "Akamai DDoS Protection.", 2019.

[Aka21] Akamai Technologies. "What does CDN stand for? CDN Definition.", 2021. [Online; accessed 15-Mar-2021].

[ALS13] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. "More efficient oblivious transfer and extensions for faster secure computation." In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 535–548. ACM, 2013.

[Ano19] Anonymous authors. "A Secure Sign-On Protocol for Smart Homes over Named Data Networking." *IEEE Communications Magazine*, **57**(7):62–68, July 2019.

[App20] Apple Inc. "Sign In with Apple.", 2020. Accessed: 2020-05-23.

[ATN19] Jari Arkko, Brian Trammell, Mark Nottingham, Christian Huitema, Martin Thomson, Jeff Tantsura, and Niels ten Oever. "Considerations on Internet Consolidation and the Internet Architecture." Internet-Draft draft-arkko-iab-internet-consolidation-01, IETF Working Draft, March 2019.

[BBC09] Patrik Bichsel, Carl Binding, Jan Camenisch, Thomas Groß, Tom Heydt-Benjamin, Dieter Sommer, and Greg Zaverucha. "Cryptographic Protocols of the Identity Mixer Library." Technical Report RZ 3730, IBM Research – Zurich, 2009.

[BBC19] BBC News. "British Airways faces record £183m fine for data breach.", 2019.

[BCR16] Hitesh Ballani, Yatin Chawathe, Sylvia Ratnasamy, Timothy Roscoe, and Scott Shenker. "Off by default!" 2016.

[BDN18] Dan Boneh, Manu Drijvers, and Gregory Neven. "Compact multi-signatures for smaller blockchains." In *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 435–464. Springer, 2018.

[Ben14] Juan Benet. "Ipfs-content addressed, versioned, p2p file system." *arXiv preprint arXiv:1407.3561*, 2014.

[BGW20] Dan Boneh, Sergey Gorbunov, Riad Wahby, Hoeteck Wee, and Zhenfei Zhang. "BLS Signatures." Internet-Draft draft-irtf-cfrg-bls-signature-04, IETF Secretariat, September 2020. `https://tools.ietf.org/html/draft-irtf-cfrg-bls-signature-04`.

[BHG13] Emmanuel Baccelli, Oliver Hahm, Mesut Gunes, Matthias Wahlisch, and Thomas C Schmidt. "RIOT OS: Towards an OS for the Internet of Things." In *2013 IEEE conference on computer communications workshops (INFOCOM WKSHPS)*, pp. 79–80. IEEE, 2013.

[BHM19] R. Barnes, J. Hoffman-Andrews, D. McCarney, and J. Kasten. "Automatic Certificate Management Environment (ACME)." RFC 8555, RFC Editor, March 2019.

[Bla19] Michael Blackman. "Hospital Health IT Interoperability.", 2019.

[BLS01] Dan Boneh, Ben Lynn, and Hovav Shacham. "Short Signatures from the Weil Pairing." In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, pp. 514–532, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[Bra07] Stefan Brands. "The problem(s) with OpenID.", 2007. Accessed: 2020-05-23.

[BS04] F. Baker and P. Savola. "Ingress Filtering for Multihomed Networks." BCP 84, RFC Editor, March 2004.

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. "Ciphertext-policy attribute-based encryption." In *2007 IEEE symposium on security and privacy (SP'07)*, pp. 321–334. IEEE, 2007.

[Cal19] Californians for Consumer Privacy. "California Consumer Privacy Act (CCPA).", 2019.

[CCJ11] Eun Kyoung Choe, Sunny Consolvo, Jaeyeon Jung, Beverly Harrison, and Julie A Kientz. "Living in a glass house: a survey of private moments in the home." In *Proceedings of the 13th international conference on Ubiquitous computing*, pp. 41–44, 2011.

[Cen20] Centers for Medicare and Midcaid Services. "Data.CMS.gov.", 2020.

[CER99] CERT Coordination Center. "CERT Incident Note IN-99-04.", 1999.

[Cho02] Gihwan Cho. "Using predictive prefetching to improve location awareness of mobile information service." In *International Conference on Computational Science*, pp. 1128–1136. Springer, 2002.

[Cim17] Catalin Cimpanu. "14766 Let's Encrypt SSL Certificates Issued to PayPal Phishing Sites.", 2017. Accessed: 2020-06-01.

[Clo19] CloudFlare. "CloudFlare Advanced DDoS Attack Protection.", 2019.

[Com16] European Commission. "General Data Protection Regulation (GDPR), chapter II, Article 5.", 2016. Accessed: 2020-05-23.

[CPZ16] Jianxun Cao, Dan Pei, Xiaoping Zhang, Beichuan Zhang, and Youjian Zhao. "Fetching popular data from the nearest replica in NDN." In *2016 25th International Conference on Computer Communication and Networks (ICCCN)*, pp. 1–9. IEEE, 2016.

[DK09] Pralhad Deshpande, Anand Kashyap, et al. "Predictive methods for improved vehicular WiFi access." In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pp. 263–276. ACM, 2009.

[Doo20a] DoorDash Blog. "Important security notice about your DoorDash account.", 2020.

[Doo20b] DoorDash Blog. "Important security notice about your DoorDash account.", 2020.

[EE 13] EE Times. "MCU market turns to 32-bits and ARM." https://www.eetimes.com/mcu-market-turns-to-32-bits-and-arm/, 2013. Accessed: 2021-01-24.

[EU 19] EU GDPR.ORG. "The EU General Data Protection Regulation (GDPR).", 2019.

[FK11] S. Frankel and S. Krishnan. "IP Security (IPsec) and Internet Key Exchange (IKE) Document Roadmap." RFC 6071, RFC Editor, February 2011. http://www.rfc-editor.org/rfc/rfc6071.txt.

[FKK11] A. Freier, P. Karlton, and P. Kocher. "The Secure Sockets Layer (SSL) Protocol Version 3.0." RFC 6101, RFC Editor, August 2011. http://www.rfc-editor.org/rfc/rfc6101.txt.

[FKS15] Daniel Fett, Ralf Küsters, and Guido Schmitz. "SPRESSO: A secure, privacy-respecting single sign-on system for the web." In *22nd Conference on Computer and Communications Security*, CCS. ACM, 2015.

[FS00] P. Ferguson and D. Senie. "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing." BCP 38, RFC Editor, May 2000.

[GGK15] Steven Goldfeder, Rosario Gennaro, Harry Kalodner, Joseph Bonneau, Joshua Kroll, Edward Felten, and Arvind Narayanan. "Securing Bitcoin wallets via a new DSA/ECDSA threshold signature scheme.", 2015.

[GGN16] Rosario Gennaro, Steven Goldfeder, and Arvind Narayanan. "Threshold-Optimal DSA/ECDSA Signatures and an Application to Bitcoin Wallet Security." In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *Applied Cryptography and Network Security*, pp. 156–174, Cham, 2016. Springer International Publishing.

[GN14] Ruti Gafni and Dudu Nissim. "To social login or not login? Exploring factors affecting the decision." *Issues in Informing Science and Information Technology*, **11**(1), 2014.

[Goo12] Jason Goode. "The importance of identity security." *Computer Fraud & Security*, **2012**(1), 2012.

[GS18a] Dennis Grewe, Sebastian Schildt, et al. "ADePt: Adaptive Distributed Content Prefetching for Information-Centric Connected Vehicles." In *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, pp. 1–5. IEEE, 2018.

[GS18b] Ishu Gupta and Ashutosh Kumar Singh. "A Probabilistic Approach for Guilty Agent Detection using Bigraph after Distribution of Sample Data." *Procedia Computer Science*, **125**:662–668, 2018.

[GS19] Ishu Gupta and Ashutosh Kumar Singh. "Dynamic threshold based information leaker identification scheme." *Information Processing Letters*, **147**:69–73, 2019.

[GTU13] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang. "DoS and DDoS in Named Data Networking." In *2013 22nd International Conference on Computer Communication and Networks (ICCCN)*, July 2013.

[GW16] Dennis Grewe, Marco Wagner, et al. "PeRCeIVE: Proactive caching in ICN-based VANETs." In *2016 IEEE Vehicular Networking Conference (VNC)*, pp. 1–8. IEEE, 2016.

[GWL06] Fei Guo, Jianmin Wang, and Deyi Li. "Fingerprinting relational databases." In *Proceedings of the 2006 ACM symposium on Applied computing*, pp. 487–492, 2006.

[Hea19] New York State Department of Health. "Hospital Inpatient Discharges (SPARCS De-Identified): 2015.", 2019.

[HG04] Mark Handley and Adam Greenhalgh. "Steps towards a DoS-resistant internet architecture." In *Proceedings of the ACM SIGCOMM workshop on Future directions in network architecture*, pp. 49–56. ACM, 2004.

[HPC10] Raju Halder, Shantanu Pal, and Agostino Cortesi. "Watermarking Techniques for Relational Databases: Survey, Classification and Comparison." *J. UCS*, **16**(21):3164–3190, 2010.

[HRS17] Andreas Haas, Andreas Rossberg, Derek L Schuff, Ben L Titzer, Michael Holman, Dan Gohman, Luke Wagner, Alon Zakai, and JF Bastien. "Bringing the web up to speed with WebAssembly." In *38th ACM SIGPLAN Conference on Programming Language Design and Implementation*, PLDI, 2017.

[HZ16] Zhiwen Hu, Zijie Zheng, et al. "Game theoretic approaches for wireless proactive caching." *IEEE Communications Magazine*, **54**(8):37–43, 2016.

[IC 15] IC Insights, Inc. "Microcontroller Sales Regain Momentum After Slump." `https://www.icinsights.com/news/bulletins/Microcontroller-Sales-Regain-Momentum-After-Slump/`, 2015. Accessed: 2021-01-24.

[IC 18a] IC Insights, Inc. "MCUs Sales to Reach Record-High Annual Revenues Through 2022." `https://www.icinsights.com/news/bulletins/MCUs-Sales-To-Reach-RecordHigh-Annual-Revenues-Through-2022/`, 2018. Accessed: 2021-01-24.

[IC 18b] IC Insights, Inc. "Microcontrollers Will Regain Growth After 2019 Slump." https://www.icinsights.com/news/bulletins/Microcontrollers-Will-Regain-Growth-After-2019-Slump/, 2018. Accessed: 2021-01-24.

[ISO15] ISO/IEC JTC 1/SC 27 Information security, cybersecurity and privacy protection. "ISO/IEC 27033-1:2015 Network Security Overview and Concepts." Technical report, ISO Standard, 2015.

[ISO19] ISO/IEC JTC 1/SC 27 Information security, cybersecurity and privacy protection. "ISO/IEC 27701:2019: Extension to ISO/IEC 27001 and ISO/IEC 27002 for privacy information management — Requirements and guidelines." Technical report, ISO Standard, 2019.

[IWS04] Blake Ives, Kenneth R Walsh, and Helmut Schneider. "The domino effect of password reuse." *Communications of the ACM*, **47**(4), 2004.

[JB14] Xiaoke Jiang and Jun Bi. "ncdn: Cdn enhanced with ndn." In *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 440–445. IEEE, 2014.

[JBS15] M. Jones, J. Bradley, and N. Sakimura. "JSON Web Signature (JWS)." RFC 7515, RFC Editor, May 2015. http://www.rfc-editor.org/rfc/rfc7515.txt.

[JK98] Zhimei Jiang and Leonard Kleinrock. "Web prefetching in a mobile environment." *IEEE Personal Communications*, **5**(5):25–34, 1998.

[Joh21] Jay Tillay Johnson. "Recommendations for Distributed Energy Resource Access Control." Technical report, Sandia National Lab.(SNL-NM), Albuquerque, NM (United States), 2021.

[JST09] Van Jacobson, Diana K Smetters, James D Thornton, Michael F Plass, Nicholas H Briggs, and Rebecca L Braynard. "Networking named content." In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, pp. 1–12, 2009.

[JWS03] Cheng Jin, Haining Wang, and Kang G Shin. "Hop-count filtering: an effective defense against spoofed DDoS traffic." In *Proceedings of the 10th ACM conference on Computer and communications security*, pp. 30–41. ACM, 2003.

[KL15] Ryangsoo Kim, Hyuk Lim, et al. "Prefetching-based data dissemination in vehicular cloud systems." *IEEE Transactions on Vehicular Technology*, **65**(1):292–306, 2015.

[KMW07] Balachander Krishnamurthy, Delfina Malandrino, and Craig E Wills. "Measuring privacy loss and the impact of privacy protection in web browsing." In *3rd symposium on Usable privacy and security*, SOUPS. ACM, 2007.

[LB16] Huichen Lin and Neil W Bergmann. "IoT privacy and security challenges for smart home environments." *Information*, **7**(3):44, 2016.

[Let20] Let's Encrypt. "Let's Encrypt.", 2020. Accessed: 2020-06-01.

[LSJ03] Yingjiu Li, Vipin Swarup, and Sushil Jajodia. "Constructing a virtual primary key for fingerprinting relational data." In *Proceedings of the 3rd ACM workshop on Digital rights management*, pp. 133–141, 2003.

[LSJ05] Yingjiu Li, Vipin Swarup, and Sushil Jajodia. "Fingerprinting relational databases: Schemes and specialties." *IEEE Transactions on Dependable and Secure Computing*, **2**(1):34–45, 2005.

[LWD04] Siyuan Liu, Shuhong Wang, Robert H Deng, and Weizhong Shao. "A block oriented fingerprinting scheme in relational database." In *International conference on information security and cryptology*, pp. 455–466. Springer, 2004.

[Mar21] MarketWatch, Inc. "IoT Microcontroller Market 2020 Global Industry Size, Opportunities, Key Vendors Analysis, Business Strategy, Future Plans, Competitive Landscape and Outlook 2023." `https://www.marketwatch.com/press-release`, 2021. Accessed: 2021-01-22.

[MAZ17] Spyridon Mastorakis, Alexander Afanasyev, and Lixia Zhang. "On the Evolution of ndnSIM: an Open-Source Simulator for NDN Experimentation." *ACM Computer Communication Review*, July 2017.

[MCC14] Ge Ma, Zhen Chen, Junwei Cao, Zhenhua Guo, Yixin Jiang, and Xiaobin Guo. "A tentative comparison on CDN and NDN." In *2014 IEEE international conference on systems, man, and cybernetics (SMC)*, pp. 2893–2898. IEEE, 2014.

[MG16] Giulia Mauri, Mario Gerla, et al. "Optimal content prefetching in NDN vehicle-to-infrastructure scenario." *IEEE Transactions on Vehicular Technology*, **66**(3):2513–2525, 2016.

[MGS10a] Erika McCallister, Tim Grance, and Karen Scarfone. *Guide to protecting the confidentiality of personally identifiable information.* National Institute of Standards and Technology (NIST) Special Publication 800-122, 2010.

[MGS10b] Erika McCallister, Tim Grance, and Karen Scarfone. *Guide to protecting the confidentiality of personally identifiable information.* National Institute of Standards and Technology (NIST) Special Publication 800-122, 2010.

[Ndn21] Ndn-cxx Contributors. "ndn-cxx: NDN C++ library with eXperimental eXtensions." `https://github.com/named-data/ndn-cxx`, 2021. Accessed: 2021-01-21.

[Neu19] Neustar. "Neustar Defense and Performance.", 2019.

[NFD21] NFD Contributors. "NFD - Named Data Networking Forwarding Daemon." `https://github.com/named-data/nfd`, 2021. Accessed: 2021-01-21.

[NN08] Anthony J Nicholson and Brian D Noble. "Breadcrumbs: forecasting mobile connectivity." In *Proceedings of the 14th ACM international conference on Mobile computing and networking*, pp. 46–57. ACM, 2008.

[Nor21] Nordic Semiconductor Inc. "Nordic nRF52840." `https://www.nordicsemi.com/Products/Low-power-short-range-wireless/nRF52840`, 2021. Accessed: 2021-01-26.

[OBM18] Aleksandr Ometov, Sergey Bezzateev, Niko Mäkitalo, Sergey Andreev, Tommi Mikkonen, and Yevgeni Koucheryavy. "Multi-factor authentication: A survey." *Cryptography*, **2**(1):1, 2018.

[Orm03] H. Orman. "The Morris worm: a fifteen-year perspective." *IEEE Security & Privacy*, **1**(5):35–43, 2003.

[OSZ20] Eric Osterweil, Angelos Stavrou, and Lixia Zhang. "21 Years of Distributed Denial-of Service: Current State of Affairs." *Computer*, **53**(7):88–92, 2020.

[PG09] Panagiotis Papadimitriou and H. Garcia-Molina. "A Model for Data Leakage Detection." In *2009 IEEE 25th International Conference on Data Engineering*, pp. 1307–1310, March 2009.

[PG11] P. Papadimitriou and H. Garcia-Molina. "Data Leakage Detection." *IEEE Transactions on Knowledge and Data Engineering*, **23**(1):51–63, Jan 2011.

[prn18] prnewswire.com. "The DDoS protection and mitigation market size is expected to grow from USD 1.94 billion in 2018 to USD 4.10 billion by 2023, at a Compound Annual Growth Rate (CAGR) of 16.1." https://www.prnewswire.com/news-releases/the-ddos-protection-and-mitigation-market-size-is-expected-to-grow-from-usd-1-94-billion-in-2018-to-usd-4-10-billion-by-2023–at-a-compound-annual-growth-rate-cagr-of-16-1-300645633.html, 2018.

[Pro21] The Ethereum Project. "Ethereum 2.0 Specifications." `https://github.com/ethereum/eth2.0-specs`, 2021.

[PS16] David Pointcheval and Olivier Sanders. "Short randomizable signatures." In *Cryptographers' Track at the RSA Conference*. Springer, 2016.

[PS18] David Pointcheval and Olivier Sanders. "Reassessing security of randomizable signatures." In *Cryptographers' Track at the RSA Conference*, pp. 319–338. Springer, 2018.

[Rab05] Michael O Rabin. "How To Exchange Secrets with Oblivious Transfer." *IACR Cryptology ePrint Archive*, **2005**:187, 2005.

[Res18] E. Rescorla. "The Transport Layer Security (TLS) Protocol Version 1.3." RFC 8446, RFC Editor, August 2018.

[RL96] Ronald L Rivest and Butler Lampson. "SDSI-a simple distributed security infrastructure." 1996.

[Ros14] Christian Rossow. "Amplification Hell: Revisiting Network Protocols for DDoS Abuse." In *NDSS*, 2014.

[RST11] Andre B Reis, Susana Sargento, and Ozan K Tonguz. "On the performance of sparse vehicular networks with road side units." In *2011 IEEE 73rd Vehicular Technology Conference (VTC Spring)*, pp. 1–5. IEEE, 2011.

[RZ13] Ying Rao, Huachun Zhou, et al. "Proactive caching for enhancing user-side mobility support in named data networking." In *2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*, pp. 37–42. IEEE, 2013.

[SAN21] SANS Institute. "SANS: Network Security Resources.", 2021. [Online; accessed 15-Mar-2021].

[SG20] Ashutosh Kumar Singh and Ishu Gupta. "Online information leaker identification scheme for secure data sharing." *Multimedia Tools and Applications*, pp. 1–18, 2020.

[Shi21] Mitsunari Shigeo. "BLS threshold signature." `https://github.com/herumi/bls`, 2021.

[Sim20] SimilarTech.com. "Market share & web usage statistics: OpenID.", 2020. Accessed: 2020-05-23.

[SJ09] Diana Smetters and Van Jacobson. "Securing network content." Technical report, PARC, 2009.

[SKS20] Yumi Sakemi, Tetsutaro Kobayashi, Tsunekazu Saito, and Riad Wahby. "Pairing-Friendly Curves." Internet-Draft draft-irtf-cfrg-pairing-friendly-curves-09, IETF Secretariat, November 2020. `http://www.ietf.org/internet-drafts/draft-irtf-cfrg-pairing-friendly-curves-09.txt`.

[SLC19] Manu Sporny, Dave Longley, and David Chadwick. "Verifiable Credentials Data Model 1.0." Technical Report W3C Recommendation 19 November 2019, W3C, November 2019.

[Sma15] SmartThingsCommunity Contributors. "turn-it-on-when-it-opens." `https://github.com/SmartThingsCommunity/SmartThingsPublic/blob/master/smartapps/smartthings/turn-it-on-when-it-opens.src/turn-it-on-when-it-opens.groovy`, 2015. Accessed: 2021-01-24.

[Sma20] SmartThingsCommunity Contributors. "Example code in SmartThings SDK for Direct Connected Devices for C." `https://github.com/SmartThingsCommunity/st-device-sdk-c/blob/master/example/example.c`, 2020. Accessed: 2021-01-24.

[Sma21] SmartThingsCommunity Contributors. "SmartThingsCommunity GitHub." `https://github.com/SmartThingsCommunity`, 2021. Accessed: 2021-01-24.

[Squ] "Squid Optimising Web Delivery." `http://www.squid-cache.org/`. Accessed: 2020-07-26.

[SS75] Jerome H Saltzer and Michael D Schroeder. "The protection of information in computer systems." *Proceedings of the IEEE*, **63**(9):1278–1308, 1975.

[SWA17] Wentao Shang, Zhehao Wang, Alexander Afanasyev, Jeff Burke, and Lixia Zhang. "Breaking out of the cloud: Local trust management and rendezvous in Named Data Networking of Things." In *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation*, pp. 3–13, 2017.

[Tar08] Gábor Tardos. "Optimal probabilistic fingerprint codes." *Journal of the ACM (JACM)*, **55**(2):1–24, 2008.

[The14] The Smart Grid Interoperability Panel – Smart Grid Cybersecurity Committee. "NISTIR 7628 Rev. 1: Guidelines for Smart Grid Cybersecurity." Technical report, NIST, 2014.

[The19a] The New York Times. "Capital One Data Breach Compromises Data of Over 100 Million.", 2019.

[The19b] The Wall Street Journal. "Amazon Investigates Employees Leaking Data for Bribes.", 2019.

[TST17] Samuel Tweneboah-Koduah, Knud Erik Skouby, and Reza Tadayoni. "Cyber security threats to IoT applications and service domains." *Wireless Personal Communications*, **95**(1):169–185, 2017.

[UJS13] Blase Ur, Jaeyeon Jung, and Stuart Schechter. "The current state of access control for smart devices in homes." In *Workshop on Home Usable Privacy and Security (HUPS)*, volume 29, pp. 209–218. HUPS 2014, 2013.

[Wik21a] Wikipedia contributors. "Forward secrecy.", 2021. [Online; accessed 15-Mar-2021].

[Wik21b] Wikipedia contributors. "Network Security.", 2021. [Online; accessed 15-Mar-2021].

[XF09] Hequn Xian and Dengguo Feng. "Leakage identification for secret relational data using shadowed watermarks." In *2009 International Conference on Communication Software and Networks*, pp. 473–478. IEEE, 2009.

[YAC15] Yingdi Yu, Alexander Afanasyev, David Clark, kc claffy, Van Jacobson, and Lixia Zhang. "Schematizing Trust in Named Data Networking." In *Proc. of ACM ICN*, 2015.

[Yas20] Jeffrey Yasskin. "Signed HTTP Exchanges." Internet-Draft draft-yasskin-http-origin-signed-responses-09, IETF Secretariat, July 2020. `http://www.ietf.org/internet-drafts/draft-yasskin-http-origin-signed-responses-09.txt`.

[Yas21] Jeffrey Yasskin. "Use Cases and Requirements for Web Packages." Internet-Draft draft-yasskin-wpack-use-cases-02, IETF Secretariat, April 2021. `https://www.ietf.org/archive/id/draft-yasskin-wpack-use-cases-02.txt`.

[YWA05] Xiaowei Yang, David Wetherall, and Thomas Anderson. "A DoS-limiting network architecture." In *ACM SIGCOMM Computer Communication Review*, volume 35, pp. 241–252. ACM, 2005.

[ZAB14] Lixia Zhang, Alexander Afanasyev, Jeffrey Burke, Van Jacobson, Patrick Crowley, Christos Papadopoulos, Lan Wang, Beichuan Zhang, et al. "Named data networking." *ACM SIGCOMM Comp. Comm. Review*, 2014.

[ZAZ17] Zhiyi Zhang, Alexander Afanasyev, and Lixia Zhang. "NDNCERT: Universal Usable Trust Management for NDN." In *Proceedings of the 4th ACM Conference on Information-Centric Networking*, ICN '17, pp. 178–179, New York, NY, USA, 2017. ACM.

[ZGM20] Zhiyi Zhang, Yu Guan, Xinyu Ma, Tianyuan Yu, and Lixia Zhang. "Sovereign: User-Controlled Smart Homes." *arXiv preprint arXiv:2006.06131*, 2020.

[ZKS21] Zhiyi Zhang, Michał Król, Alberto Sonnino, Lixia Zhang, and Etienne Rivière. "EL PASSO: Efficient and Lightweight Privacy-preserving Single Sign On." *Proceedings on Privacy Enhancing Technologies*, **2021**(2):70–87, 2021.

[ZLD19] Zhiyi Zhang, Tianxiang Li, John Dellaverson, and Lixia Zhang. "RapidVFetch: Rapid Downloading of Named Data via Distributed V2V Communication." In *2019 IEEE Vehicular Networking Conference (VNC)*, pp. 1–8. IEEE, 2019.

[ZS20] Zhiyi Zhang and Junxiao Shi. "NDNCERT Specification 0.3." https://github.com/named-data/ndncert/wiki/NDNCERT-Protocol-0.3, 2020. Accessed: 2020-06-01.

[ZVM19] Zhiyi Zhang, Vishrant Vasavada, Xinyu Ma, and Lixia Zhang. "DLedger: An IoT-Friendly Private Distributed Ledger System Based on DAG." *CoRR*, **abs/1902.09031**, 2019.

[ZVO19] Zhiyi Zhang, Vishrant Vasavada, Eric Osterweil, Lixia Zhang, et al. "Expect more from the networking: Ddos mitigation by fitt in named data networking." *arXiv preprint arXiv:1902.09033*, 2019.

[ZXG20] Zhiyi Zhang, Guorui Xiao, Yu Guan, Xinyu Ma, and Lixia Zhang. "ReLiShare: Reliable Leaker Identification in Sensitive Data Sharing.", 2020.

[ZYR18] Zhiyi Zhang, Yingdi Yu, Sanjeev Kaushik Ramani, Alex Afanasyev, and Lixia Zhang. "NAC: Automating access control via Named Data." In *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*, pp. 626–633. IEEE, 2018.

[ZYZ18] Zhiyi Zhang, Yingdi Yu, Haitao Zhang, Eric Newberry, Spyridon Mastorakis, Yanbiao Li, Alexander Afanasyev, and Lixia Zhang. "An Overview of Security Support in Named Data Networking." *IEEE Communications Magazine*, **56**(11):62–68, November 2018.