



NTNU – Trondheim
Norwegian University of
Science and Technology

Security Properties of Information-centric Networks

Haakon Garseg Mørk

Master of Science in Communication Technology

Submission date: June 2015

Supervisor: Stig Frode Mjøl̄snes, ITEM

Norwegian University of Science and Technology
Department of Telematics

Title: Security Properties of Information-centric Networking
Student: Haakon Garseg Mørk

Problem description:

The translation from name to address and location is a fundamental problem to all networks. Named Data Networking (NDN) is a proposal for content-centric discovery and routing approach to networking going on at the University of California, Los Angeles (UCLA) [1], which is part of the inspiration and a contact point for this work.

This project aims to understand the ideas and concepts of named data networking, and investigate the potential these notions hold for network security, even for properties of anonymity and privacy. The student work should be to build a public key distribution application that uses ChronoSync [2] and runs over NDN. This approach will allow for easy experimentation in the regular internet protocols environment.

[1] UCLA. Named Data Networking. Web site at <http://named-data.net/>

[2] UCLA, Named-data Networking. ChronoSync <https://github.com/named-data/ChronoSync>

Responsible professor: Stig F. Mjølshes
Supervisor: Stig F. Mjølshes

Abstract

The IP network was built decades ago, and with today's use of Internet, a new network layer protocol is much needed. Named Data Networking (NDN) is a proposal for content-centric discovery and routing. Yet, the public key infrastructure issue has not been solved in NDN. Identity-based cryptography (IBC) seems to be applicable to wireless sensor networks, and even more applicable when deployed over NDN.

In this paper I will explain the NDN architecture and the basics of IBC. Further, I will model and implement a trust model in a thought sensor network using IBC, running over NDN.

Implementing and testing my proposal verifies the relevancy of IBC over wireless sensor network running over NDN, and the usability of developing applications over NDN.

I formally and informally prove the security in the protocols suggested for device registration and data pull under deployment in the application.

Sammendrag

IP nettverket ble bygd for flere tiår siden, og med dagens bruk av Internet ser vi at en ny nettverksprotokoll er sårt trengt. Named Data Networking (NDN) er en foreslått nettverksprotokoll som baserer seg på innhold, istedenfor punkt-til-punkt arkitekturen som er grunnlaget for IP. Selv med flere tiårs bruk av Internet, er ennå ikke problemene med Public Key Infrastructure (PKI) løst. I NDN, har man heller ikke klart å finne en løsning på dette. Identitetsbasert kryptografi (IBC) viser seg å være anvendelig til trådløse sensornettverk, og enda mer når sensornettverket kommuniserer over NDN.

I denne masteroppgaven forklarer jeg NDN arkitekturen og de grunnleggende prinsippene i IBC. Jeg modellerer og implementerer en applikasjon for å demonstrere bruken av IBC over NDN i et tenkt sensornettverk.

Implementasjonen og testingen av mitt bidrag verifiserer relevansen av IBC i et sensornettverk som kjører over NDN, samt brukervennligheten rundt det å utvikle applikasjoner over NDN.

Jeg formelt og uformelt beviser sikkerheten i protokollene som er foreslått til enhetsregistrering og dataforespørsel i applikasjonen.

Acknowledgments

I would like to thank Stig Frode Mjølsnes for guidance and valuable suggestions throughout the process. Also, I would like to thank Lixia Zhang at the University of California, Los Angeles (UCLA), for introducing me to Named Data Networking (NDN).

Eirik Auran Rathe contributed with editorial advice and suggestions.

Contents

List of Figures	xi
Listings	xiii
List of Tables	xv
List of Terms	xvii
List of Acronyms	xix
1 Introduction	1
1.1 Motivation	1
1.2 Problem and Scope	1
1.3 Methodology	2
1.4 Outline	2
2 Background	3
2.1 Motivation for Information-Centric Networking	3
2.2 Content Centric Network & Named Data Network	5
2.3 NDN Architecture	5
2.3.1 Brief Introduction	5
2.3.2 NDN - Based on Existing Concepts	6
2.3.3 Packets	6
2.3.4 Names	7
2.3.5 Network Node	8
2.3.6 Incoming Interest	11
2.3.7 Incoming Data	12
2.3.8 Security	13
2.3.9 Attacks	16
2.4 Related work	16
3 Identity-Based Cryptography	19
3.1 Notations	19

3.2	Concept	19
3.3	Security	22
4	Key Distribution Scheme	25
4.1	File Synchronization Module	25
4.2	Key Distribution	26
4.3	Key Revocation	26
5	Sensor Application	29
5.1	Health Sensors	29
5.2	Health Sensor System	30
5.2.1	Rendezvous Authentication	30
5.2.2	Threat Model	31
5.2.3	System Initialization Phase	31
5.2.4	Device Registration Phase	32
5.2.5	Deployment Phase	35
5.2.6	Key Distribution using File Synchronization Module	37
5.3	Informal Security Analysis	37
5.3.1	Access Control	37
5.3.2	Confidentiality	38
5.3.3	Integrity and Authenticity	38
5.3.4	Availability	38
5.3.5	Trust Model	38
6	Implementation and Testing	41
6.1	Installing Named Data Networking Forwarding Daemon	41
6.2	Installing PyNDN2	42
6.3	Installing Identity-Based Cryptography	42
6.4	File Synchronization Module - Implementation	42
6.5	Health Sensor System - Implementation	43
6.5.1	Key Storage	43
6.5.2	Name Structure	44
6.5.3	Access Control	45
6.5.4	Packet Design	45
6.5.5	Running the Code	46
6.6	Testing	48
6.6.1	Computers	48
6.6.2	Key Sizes	48
6.6.3	Performance	48
6.7	NDN Testbed	50
7	Discussion	53

7.1	Named Data Networking	53
7.2	Identity-Based Cryptography in Named Data Networking	54
7.3	Scalability	55
7.4	Preshared Secret	56
7.5	Key Distribution	56
7.6	Testing	57
7.7	Other Use Cases	57
8	Conclusion and Future Work	59
8.1	Conclusion	59
8.2	Future Work	60
	References	61
	Appendices	
A	ChronoSync	67
B	Formal Security Analysis Code	69
B.1	Scyther Security Analysis of Device Registration	69
B.2	Scyther Security Analysis of Data Pull	71

List of Figures

2.1	Peak Period Aggregate Traffic Composition	4
2.2	NDN packets	6
2.3	Model of IP node	9
2.4	Model of NDN node	10
2.5	Multicast in NDN	11
2.6	Incoming Interest - NDN node	12
2.7	Incoming Data - NDN node	13
2.8	NDN versus IP architecture	14
3.1	IBC methods	21
4.1	File Synchronization in NDN	27
4.2	File Synchronization Module	28
5.1	Health Sensor System	30
5.2	Device Registration, phase 1	33
5.3	Device Registration, phase 2	34
5.4	Data pull under deployment	36
6.1	Class topology	44
6.2	Init Interest and Data packet	46
6.3	Sensor Interest and Data packet	47
6.4	HSS testing - computer topology	50
6.5	NDN Testbed map	51

Listings

6.1	Start NFD	47
6.2	Start PKG	47
6.3	Start a device registering a prefix.	47
6.4	Start a device that will express Interest in Data	48
B.1	Device Registration SPDL	69
B.2	Data Pull SPDL	71

List of Tables

2.1	Bandwidth comparison of NDN and IP	11
3.1	IBC Notations	19
6.1	Test computers	48
6.2	Size chart of IBC parameters	49
6.3	Time chart of cryptographic computations	49
6.4	Round trip time of protocols	50
7.1	Comparison with PKC and IBC	55
B.1	SPDL mapping device registration	69
B.2	SPDL mapping data pull	71

List of Terms

Data	is the packet sent in response to an Interest .
\mathbb{G}	group of prime order.
global passive adversary	is an adversary which can monitor the entire network flow.
host	is a node that holds the relevant Data .
Interest	is the packet sent from a node to another, requesting some content with a Name .
multicast	is a one-to-many or many-to-many distribution of Data .
Name	is the name of content related to a Data packet in NDN.
node	is referred to as a machine participating in a NDN network.
p	prime order.
publisher	is the owner/producer of the Data , and is required to signed its Data .
requester	is a node that has expressed an Interest to some Data .

simple assumption	is defined through a security game in which an adversary first gets a challenge whose size only depends on the security parameter, and must then output a unique solution without further interaction [HKS15].
sink node	is a stationary base station which acts as a gateway for all sensors in a WSN [PS12, Section 1.1].
synchronization group	is group of nodes that synchronizes with a given Name .

List of Acronyms

ACL Access Control List.

AES Advanced Encryption Standard.

API Application Programming Interface.

ARP Address Resolution Protocol.

ASPs Application Service Providers.

BAS Building Automation System.

BDH Bilinear Diffie-Hellman Problem.

BMS Building Management System.

CCN Content Centric Networking.

CDN Content Distribution Network.

CEK Content-Encryption Key.

CGM Continuous Glucose Monitor.

CIA Confidentiality, Integrity and Availability.

CS Content Store.

DBDH Decisional Bilinear Diffie-Hellman Problem.

DDoS Distributed Denial of Service.

DNS Domain Name System.

DNSSec DNS Security.

DOI Digital Object Identifier.

DoS Denial of Service.

ECC Elliptic Curve Cryptography.

FIA Future Internet Architecture.

FIB Forwarding Information Base.

FSM File Synchronization Module.

HIBC Hierarchical Identity-Based Cryptography.

HLR Home Location Register.

HSS Health Sensor System.

IBC Identity-Based Cryptography.

IBE Identity-Based Encryption.

IBS Identity-Based Signature.

ICN Information-Centric Networking.

ID Identity.

IoT Internet of Things.

IP Internet Protocol.

IPsec Internet Protocol Security.

IPv4 IP version 4.

IPv6 IP version 6.

IRTF Internet Research Task Force.

KB Kilobyte.

KDC Key Distribution Center.

LAN Local Area Network.

MACAW Media Access Protocol for Wireless LANs.

MITM Man In The Middle.

MPK Master Public Key.

MSK Master Secret Key.

NDN Named Data Networking.

ndn-cxx Named Data Networking C++ library with eXperimental eXtension.

NFC Near Field Communication.

NFD Named Data Networking Forwarding Daemon.

NSF National Science Foundation.

NTNU Norwegian University of Science and Technology.

PARC Palo Alto Research Center.

PBC Pairing-Based Cryptosystems.

PIT Pending Interest Table.

PK Public Key.

PKG Private Key Generator.

PKI Public Key Infrastructure.

PyNDN2 Named Data Networking Client Library in Python.

RIB Routing Information Base.

RSVP Resource ReSerVation Protocol.

SDSI Simple Distributed Security Infrastructure.

SHA1 Secure Hash Algorithm 1.

SK Secret Key.

SPDL Security Protocol Description Language.

SSN Social Security Number.

TCP Transmission Control Protocol.

TLS Transport Layer Security.

TTP Trusted Third Party.

UCLA University of California, Los Angeles.

UDP User Datagram Protocol.

URI Uniform Resource Identifier.

URL Uniform Resource Locator.

WSN Wireless Sensor Network.

Chapter 1

Introduction

1.1 Motivation

The translation from name to address and location is a fundamental problem to all networks. Named Data Networking (NDN) is a proposal for content-centric discovery and routing approach to networking going on at the University of California, Los Angeles (UCLA), which is part of the inspiration and a contact point for this work.

In general, the name to address resolution can either be maintained by a catalogue lookup service, such as Domain Name System (DNS) (Internet) and Home Location Register (HLR) (mobile networks), or resolved on-the-fly by a protocol on request, such as Address Resolution Protocol (ARP) (Local Area Network (LAN)). There has been done a tremendous amount of work on the naming problem in distributed systems, some became big failures (e.g. X.500) others such as the web Uniform Resource Locator (URL)s are very successful. Bringing things even further, the Digital Object Identifier (DOI) system is a Uniform Resource Identifier (URI) directed at the content/object itself rather than a location. Very much related to the name/address problem is the information security problem of efficient and practical public key distribution, which remain unsolved in practice, even though a significant number of digital certificate and verification protocols and schemes have been proposed, and systems tested over the last two decades. One notable and early theoretical proposal is Adi Shamir's Identity-Based Cryptography (IBC) proposal [Sha84], and subsequent work, that may be revisited and applicable to NDN.

1.2 Problem and Scope

Designing a new network protocol for the future Internet, one of the most significant changes should be security. Trust management plays a big part in security, and thus we cannot design trust management on known Internet Protocol (IP) failures such as X.500. Public Key Infrastructure (PKI) is a tough challenge to solve and it is

probably not a rigid solution, but rather case specific. NDN is being designed with security in mind, but the issue of trust management is yet to be solved.

In addition to the problem description, I also address the trust management issue in a thought sensor device network using IBC. By using the Named Data Networking Forwarding Daemon (NFD) I will implement my proposal for such sensor network over NDN, and contribute with ideas and concepts around such network.

1.3 Methodology

When developing applications over NDN it is important to understand the architecture of the protocol. I will study the protocol, reading the guide for NDN developers [ASZ⁺15] and papers published by the NDN-team¹ describing protocol features in addition to existing applications running over NDN.

The concept of IBC should be thoroughly understood, as I will apply it in my application. Thus I will study IBC, finding relevant papers at Google Scholar².

Establishing a solid background knowledge in these topics, I first design the application flow in sequence diagrams. Based on the Application Programming Interface (API) to the Named Data Networking Client Library in Python (PyNDN2), I try to implement the proposed design and see where changes can be made to minimize communication overhead, maximize security (i.e. Confidentiality, Integrity and Availability (CIA)) and maximize usability. The implementation will be tested and cryptographic parameters will be measured. I will prove the security in the protocols I propose and finally I will discuss the work done.

1.4 Outline

This thesis will first introduce NDN, one of the proposed protocols for the future Internet. I will explain the architecture of NDN as well as some related work regarding my application proposal and IBC. The concept of IBC will also be explained. The specifications for the Health Sensor System (HSS) application will be explained in detail and implementation choices will be discussed. I will present the results of the implementation and testing. At last, a discussion around the research topics in the thesis will be presented and finally my conclusion around the same topics.

¹NDN Publications - <http://named-data.net/publications/>

²Google Scholar - <https://scholar.google.no/>

Chapter 2

Background

"We model the future on the past. Sometimes that's a mistake."

— Van Jacobsen, *SIGCOMM 2001*

This chapter will give a brief overlook of the motivation for Information-Centric Networking (ICN), as well as explaining details of the ICN protocol NDN. The NDN architecture will be reviewed. Finally there will be a quick summary of related work to this thesis.

2.1 Motivation for Information-Centric Networking

When Internet was created in the 1960's, the researchers where inspired by the existing communication network; the telecommunication network. Because it was natural and logical to think that people would send and receive short messages and instructions, the point-to-point communication model was a logical choice of architecture. As Internet has developed, the traffic has increased enormously over the past few years. In the Global Internet Phenomena Report 1H2014 done by Sandvine [San14], shows that close to 64% of all IP traffic in North America was Real-Time Entertainment streaming. In Figure 2.1 it can easily be seen that most of the traffic is content download, and not communication as Internet originally was designed for. With this in mind, the IP architecture does not provide an efficient transport model for what we are actually using the network for.

When designing the IP network, security was not the first priority. A logical thought considering that they did not know what the Internet is being used for nowadays and how big it has become. Many protocols related to Internet has been designed and deployed mainly with the goal of functionality, not thinking about security. In the years after the birth of Internet it was discovered that Internet needed security at several layers, due to the increase of application requirements and transmission importance. Internet Protocol Security (IPsec) is a very good example of work trying to patch up security flaws in the design of Internet.

4 2. BACKGROUND

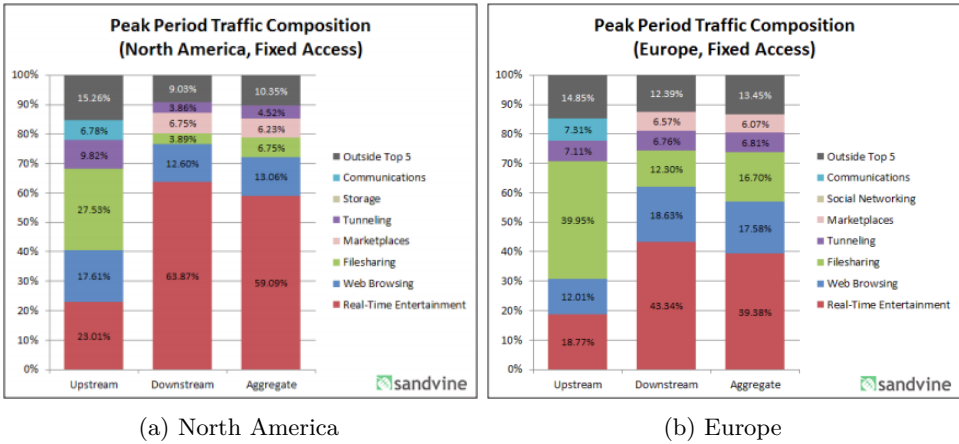


Figure 2.1: Peak Period Aggregate Traffic Composition in (a) North America and (b) Europe, fixed Access [San14]. In North America, Netflix alone is responsible for 34.21% of all content download.

Today, WiFi is disseminated across homes and buildings in many countries. Wireless technology has grown rapidly and it is predicted continuous growth in the years to come [Ros14]. The Internet of Things (IoT) trend is coming and the IP network is not designed for broadcast. Therefore wireless connection is not as easy as it should and could be. Devices should easily be able to communicate directly with each other without having to interconnect through a router.

Another problem is the network redundancy. Looking at Figure 2.1 and reading [San14] where it comes to light that Netflix stands for 34.21% of all content download in America, one can conclude that there are a lot of movies downloaded from x users geographically located close. And thus the network path from the source (e.g. Netflix) to this geographical place is allocated x too many times. This is because a node in an IP network does not know *what* it processes, but rather the packet's endpoints, i.e. *where* it goes and *where* it comes from, hence the node throws all packets after use. The fact the every node knows nothing about the content the process, makes every node dumb. The network is designed for redundancy when it comes to content download.

These design failures are some the reasons why the research for the future Internet began. ICN [ADI⁺12] is a concept developed under this research. It is built upon delivery of content, rather than the point-to-point model we previously have seen in IP. ICNs goal is to build an infrastructure of a new Internet that can achieve efficient, secure and reliable distribution of content. In 2012 Internet Research Task Force (IRTF) established ICN working group.

2.2 Content Centric Network & Named Data Network

The first network protocol purposed for ICN, Content Centric Networking (CCN), was presented by Van Jacobsen at a Google Talk in 2006. He, amongst other contributors of CCN, has been working on developing the Internet as we know it since the early start. Jacobsen has contributed to Transmission Control Protocol (TCP)/IP with his flow control algorithms [Jac88] and TCP header compression [Jac90]. CCN focuses on naming content, instead of naming IP-addresses. The research project is lead by Palo Alto Research Center (PARC). A branch of CCN is the NDN [ZAB⁺14] research project started in 2010, which Jacobsen also has contributed to. One of the biggest contributors is UCLA, with Lixia Zhang in the lead. Zhang is known for her contribution to, amongst many other, Resource ReSerVation Protocol (RSVP) [BZBH97] and Media Access Protocol for Wireless LANs (MACAW) [BDSZ94]. The NDN project is also one of few projects funded by National Science Foundation (NSF) in their Future Internet Architecture (FIA) program [Fou].

2.3 NDN Architecture

Since the knowledge of how NDN works is not disseminated amongst computer scientists, it is essential for this thesis to describe how it works. This section will describe the basic architecture of NDN [ASZ⁺15] and compare some solutions with the equivalent solutions in IP.

2.3.1 Brief Introduction

In NDN there are two types of packets, **Interest** and **Data** packet. All **Data** has been given a content **Name** by its publisher. To publish some content to the network, a user has to register the prefix, i.e. announcing the contents **Name**, which tells the network that the content can be retrieved on the announced **Name**. The retrieval can only be achieved if someone expresses an **Interest** to the content **Name**. If a user expresses **Interest** in the **Name**, the network will route the **Interest** to the closest node that holds the **Data**. The **Data** packet can be retrieved from any node, trusted or not, over any type of communication channel, secure or not. This is due to that the network layer demands a signature from the content's publisher. Finally the requester will receive the **Data** packet containing the content requested. The **Data** can be verified by the requester ensuring that the publisher who "owns" the content actually is the owner, because its cryptographically signed. If confidentiality is needed, the **Data** can be encrypted, thus the communication channel does not need to be secured.

A NDN node is slightly different to an IP node. Since the **Data** is linked to a **Name** and signed by its publisher, the node can cache the **Data** and be able to satisfy other

Interests to the same **Data**. This provides natural multicast in the network, and will ease the network load drastically.

Securing the **Data** rather than the connection, leads to a network where one can easily verify the connection between content and its publisher.

2.3.2 NDN - Based on Existing Concepts

The goal for the network design is essentially making it more secure and applicable for content without removing the communication service that IP was designed for. Designing a new network protocol we have to look at what measurements have been done in the existing IP network to tailor it towards content sharing. As the reader might notice after reading the background material, NDN is built upon concepts that we can map to well working solutions deployed over TCP/IP. Some examples are:

- BitTorrent - The concept of sharing bits of files between peers in a network is a well-working distributed method for sharing content. Requesters do not care where the content come from, but only that the content is what is requested.
- Content Distribution Network (CDN) - Many Application Service Providers (ASPs), such as Netflix and YouTube, have found out that their service performs a lot better for their costumers if they cache up their data close to where the users is located.

2.3.3 Packets

There are two types of packets in NDN; *Interest packet* and the corresponding answer, i.e. the *Data packet*, illustrated in Figure 2.2.

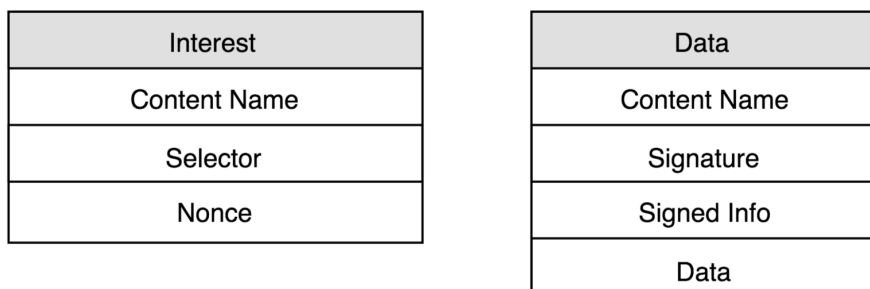


Figure 2.2: **Interest** packet and **Data** packet. Reconstructed from [Jac09].

The **Interest** packet specifies a content **Name**. The **Name** can have a hierarchical structure and signatures can be added after the URI, e.g. “/ndn/no/ntnu/haakon/-file/1/<signature>”. An **Interest** can also contain a set of different Selectors to specify original requirements for the **Data** response. Some of the Selector fields are:

- **KeyLocator** - can be used to specify where the Public Key for the signature can be found.
- **Exclude** - can be used to specify a list or a range of names that should be excluded from the **Name**. I.e. if the **Name** is “/ndn/no/ntnu” and the **Exclude** contains “/item”, the returned **Data** cannot contain “/ndn/no/ntnu/item”.
- **MustBeFresh** - if True, a node cannot answer with a **Data** packet where the **FreshnessPeriod** has expired. **FreshnessPeriod** is a time value of how long some **Data** is fresh.
- **ChildSelector** - can be used to select either the leftmost (least) or the rightmost (greatest) child, e.g. content version.
- **Min/MaxSuffixComponents** - refers to **Name** components that occur in the matching **Data** beyond the prefix.

The **Nonce** field sets automatically. This is used to uniquely identify an **Interest** and prevent looping in the network.

The **Data** packet is a response to the **Interest** packet, and contains the content **Name** and the Content itself. It also has a **MetaInfo** field that is used to specify the **FreshnessPeriod** (milliseconds), **ContentType** and **FinalBlockId**. When someone requests a file “/ndn/no/ntnu/haakon/file/1” with an **Interest**, the response will have the same **Name**, but also containing the file.

Because a **Data** packet can only exist if there is a corresponding **Interest**, NDN is pull-based. Hence unsolicited **Data** packets will be thrown away, i.e. there is no content in the network, that is not requested from someone. This reduces unwanted traffic compared to User Datagram Protocol (UDP) in IP, and minimizes the Denial of Service (DoS) vulnerability drastically.

2.3.4 Names

In today’s Internet we are well familiar with the mapping of URL and IP addresses. This mapping, done by the DNS, eases the pain of remembering an IP version 4 (IPv4) (32-bit) address and lately also IP version 6 (IPv6) (128-bit) addresses.

In a NDN network a node does not have an address nor a **Name**. The routing of packets is done by content lookup, rather than address lookup. This means that all content in the network do have a **Name**. When an **Interest** is sent, each node asks the network: “Where can I find *a* node that can provide me with this content?”.

There is no strict rules for a **Name** in NDN. This means that a network node only routes an **Interest** based on longest prefix match. Naming is left to the application design, thus it can be customized for the applications best purpose. However the network assumes hierarchical structured names, hence routing will perform better with a hierarchical **Name** design.

For the network to perform even better, the **Interest** can append some Selectors that can help the network to decide which **Data** to retrieve and where to route. With Selectors a partially known **Name** can successfully retrieve the right **Data**. E.g. when a user want to download the newest version of some content, lets say “/ndn/no/ntnu/haakon/file/<version?>”, but do not know which version is the newest, the user can append a ChildSelector to choose the newest version, if several versions are offered.

The fact that **Data** has a **Name** makes Simple Distributed Security Infrastructure (SDSI) and IBC highly applicable to NDN. Namespace-based trust was introduced in SDSI [RL96], binding names to public keys. IBC will be explained in detail in chapter 3, and is a key topic in this thesis.

2.3.5 Network Node

It may sound like a impossible task to force todays network from IP to NDN. But looking at history, IP first ran over the telecommunication network and later established its own physical network, NDN can run over the IP network and later create its own physical network. Also, if we look at an existing model of an IP node Figure 2.3 and compare it to a NDN node Figure 2.4, we see that they look much the same. The only significant difference in hardware is the storing capacity, which becomes cheaper and cheaper each month. However, the logic behind a NDN node is a bit more complex, and thus lead to more knowledge about *what* content the node has to offer. To understand this, the following entities in a NDN node should be understood:

1. Face - A term used for generalization of different interfaces, e.g. physical like Ethernet, or overlay like TCP and UDP. A Face can also be a UNIX-domain socket for communication with a local application.
2. Pending Interest Table (PIT) - All pending or recently satisfied **Interests** are stored here, together with the incoming and outgoing Face. If a new incoming

Interest matches an entry in the PIT, the incoming Face will be added to the entry.

3. Content Store (CS) - When a node receives a **Data** packet that has the corresponding entry in the PIT, it stores the **Data** packet in CS as long as possible. The CS works like a cache for the node.
4. Forwarding Information Base (FIB) - Forwarding strategy is stored for each **Name** prefix. When a node forwards an **Interest**, it will do a longest prefix lookup in the FIB and send the **Interest** further to the best matching Face.

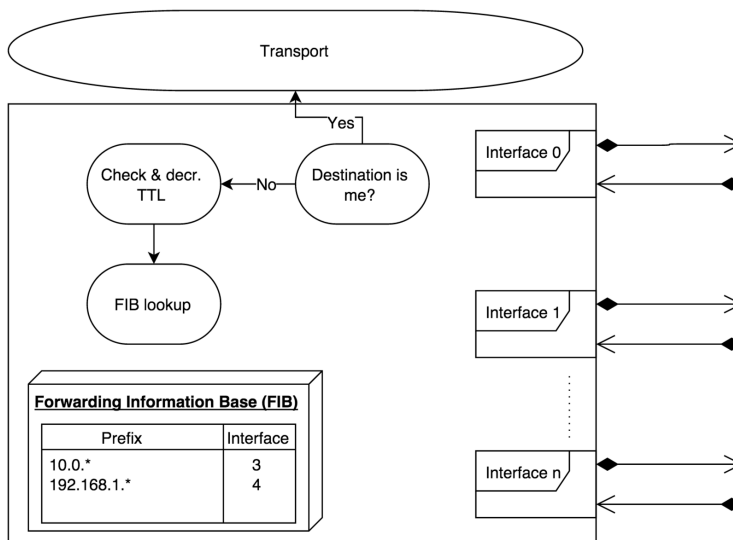


Figure 2.3: Model of IP node. A packets enters the node through an interface. The node decides whether the packet is for the node itself, or passes it further to next node, found in the FIB. Reconstructed from [Jac09].

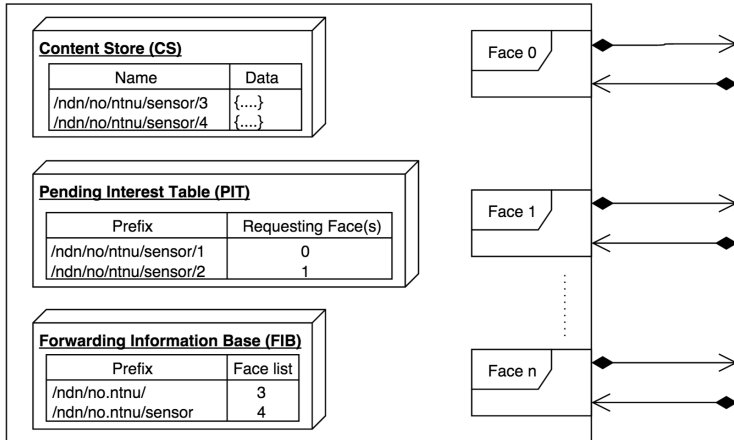


Figure 2.4: Model of NDN node. A packet enters through an Face. The node checks whether the **Interest** is already queried in the PIT, or stored in the CS, or passes it further to next node, found in the FIB. Reconstructed from [Jac09].

In contrary to an IP node, a NDN node knows *what* content comes through itself. Since all content is associated with a **Name**, a NDN node can know 1) *what* is requested, but not satisfied (i.e. stored in PIT), and 2) *what* has been satisfied earlier and still available, i.e. still cached in CS. With this knowledge the network can now satisfy **Interests** with content already stored in cache, hence the network can naturally offer multicast on network layer. Figure 2.5 illustrates a NDN network where we can see that the network does not nearly has to send equal amount of content than in an IP network. The mobile expresses an **Interest** (1) in a file named `/ntnu/file1`. The **Interest** finds its way to the publisher of the file, and thus the publisher responds with a **Data** packet (2) named `/ntnu/file1` containing the file, and the **Data** finds it way back to the mobile. When the second computer expresses the same **Interest** (3), the consecutive node has already cached the **Data** response matching to the **Interest** in its CS, hence the **Interest** is satisfied already at this point (4), and not forwarded any further. Same happens when the third computer expresses again the same **Interest** (5) to the network. Given that the file (`/ntnu/file1`) these computers are interested in is 4 gigabyte, the network saves a lot of traffic with multicast provided by NDN. Best case scenario in NDN is illustrated in Table 2.1. Every node is saving bandwidth in a NDN network compared to nodes in an IP network. Worst case scenario, where cached **Data** is thrown away, the NDN nodes will perform equal to the IP nodes.

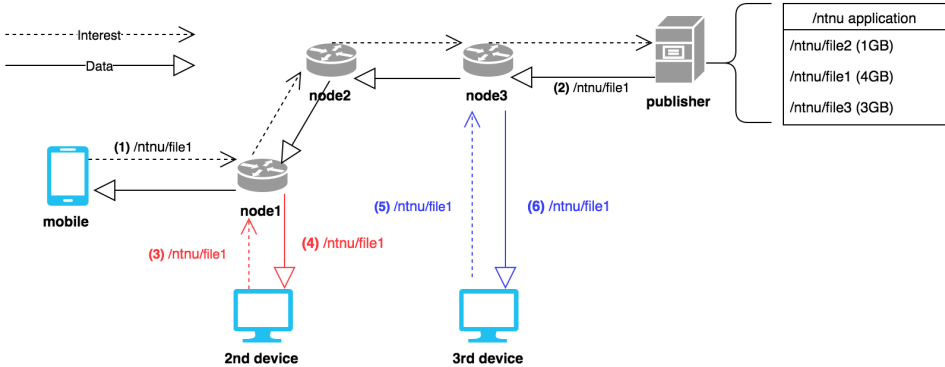


Figure 2.5: Multicast in NDN.

Node	Data in/out IP	Data in/out NDN	% bandwidth needed in NDN
node1	8GB/8GB	4GB/8GB	50%/100%
node2	8GB/8GB	4GB/4GB	50%/50%
node3	12GB/12GB	4GB/8GB	33%/66%
Publisher	-/12GB	-/4GB	-/33%

Table 2.1: Node bandwidth allocation comparison in IP and NDN (best case). Mobile, 2nddevice and 3rddevice requests `/ntnu/file1` (4GB). The NDN nodes allocates bandwidth 44.3% (incoming) and 66.3% (outgoing) compared to IP nodes.

2.3.6 Incoming Interest

In Figure 2.6 we see an incoming **Interest** through a **Face**. The node checks the **PIT** for pending or recently satisfied **Interests**. If there is no match, the node will do a lookup in **CS** to see if a corresponding **Data** packet is cached. If there is a match in the **PIT** it will only add the **Face** to the **PIT** entry. If there is a match in the **CS** the node will return the **Data**. If there is no match in either the **PIT** or the **CS** the node will make a new **PIT** entry and do a longest prefix match lookup in the **FIB** to decide which **Face(s)** to forward the **Interest**. The node waits for incoming **Data** and satisfies the **PIT** entry when the **Data** arrives, explained in subsection 2.3.7. Each **PIT** entry has its own routing strategy. I.e. whether, when, and where to forward the **Interest**.

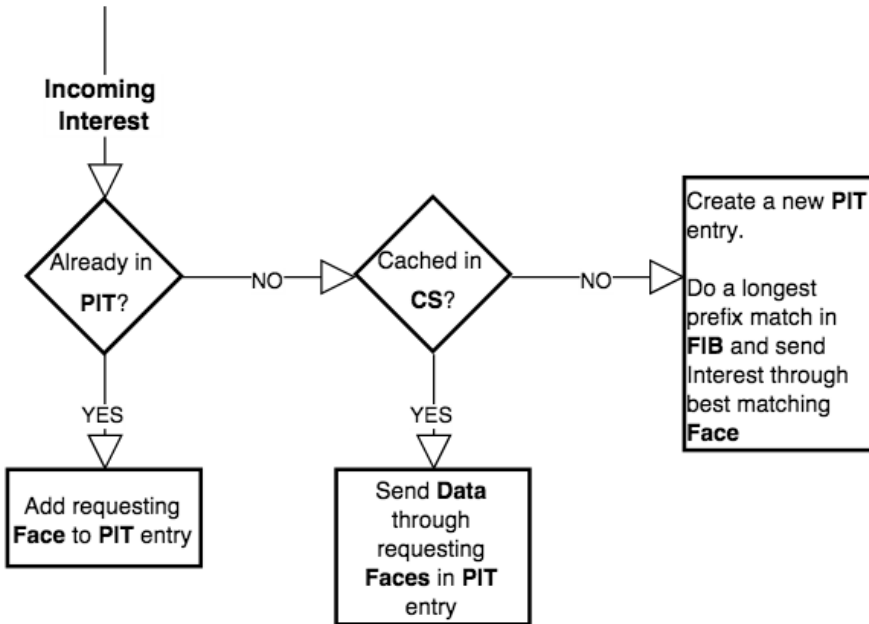


Figure 2.6: Decision tree for a NDN node when receiving an Interest.

2.3.7 Incoming Data

In Figure 2.7 we see incoming Data. The node will check the PIT for an entry, if a match is found the node will forward the Data to all the Faces registered in the PIT entry. If no match, the node will disregard the Data because it is unsolicited content. The node checks the Data from local applications cached in CS first, if there is no match, it stores the content in CS and sends the Data to all requesters (i.e. through all Faces stored in the PIT entry).

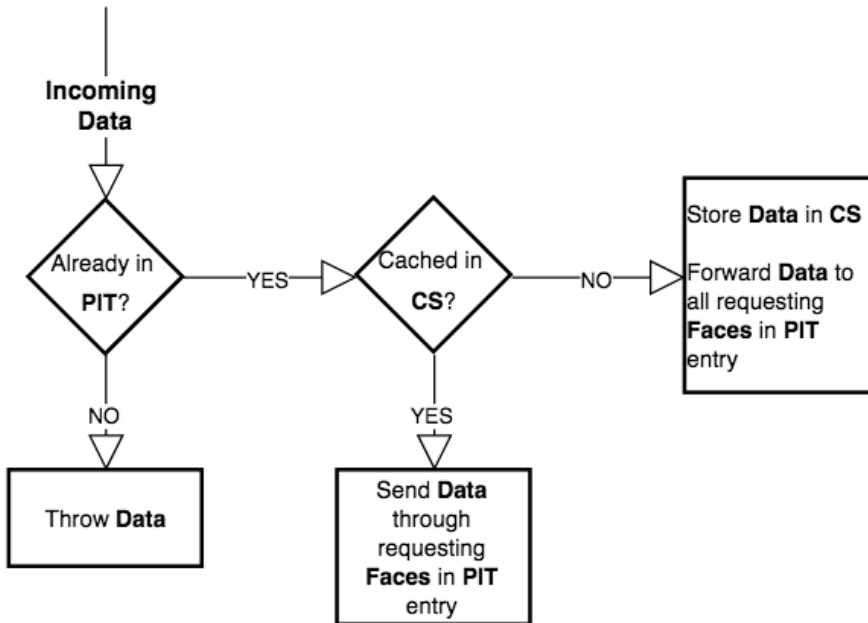


Figure 2.7: Decision tree for a NDN node when receiving Data packet.

2.3.8 Security

Below I will present why NDN facilitates good security properties, explaining some of the security aspects around NDN discussed in [SJ09], and the difference in securing data and securing channel.

Trusting Host versus Trusting Content

Doing a whole lot of mapping at different layers is not a good security model. Each mapping introduces a potentially vulnerable target for forgery. The IP network is designed in a way that makes us want to trust the host. What we are actually trusting is the mapping of the URL to the IP address. DNS points to a host address that speaks for the URL you are interested in, and thus if someone manages to forge this address exploiting DNS Security (DNSSec), you cannot tell if you talk to the right host.

As seen in Figure 2.8 the security in NDN is dealt with in one layer, rather than over several layer as needed in the IP architecture, i.e. Transport Layer Security (TLS), DNSSec, IPsec, etc.

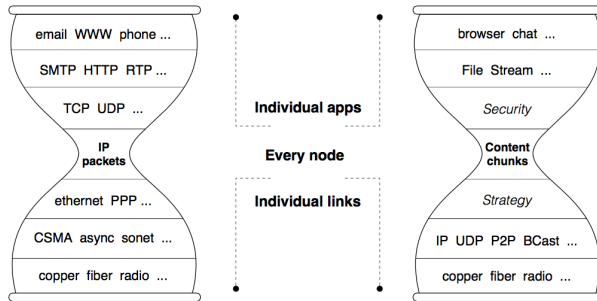


Figure 2.8: NDN versus IP architecture. The building blocks of the NDN architecture are named content chunks, in contrast to the IP architecture’s fundamental unit of communication, which is a point-to-point channel between to endpoints identified by IP addresses. [ZAB⁺14].

The content is rarely encrypted and the confidentiality is not preserved, unless there is established a secure channel using e.g. TLS. This is a problem due to the issues concerning tampering and eavesdropping. The content the host we trust provides can contain malicious software and important information can be swapped even though the channel is secured. This is the concept of securing the channel, and the trust is based on certificates. This trust is an issue itself. Due to the global PKI and essentially because the certificate is signed by a Trusted Third Party (TTP) all trust comes outside the namespace. This makes it problematic to retrieve content over IP from other sources than the trusted host because you do not trust any other than the host you are connected to via the secure channel. The host does not provide any assurance that the content is verified by the host itself, it only assures a secure channel.

A goal is to get the desired content from the intended source, unmodified in transit. Therefore a better solution would be to trust the content rather than the host of the content. This concept requires us to change the network trust. Skipping 1) all the trust based in mapping of hosts, 2) where the data comes from, and 3) securing the channel. The content should be linked to the publisher and this linkage should be signed by the publisher. The concept is to mathematically prove that the content originates from the believed publisher, and that its not modified nor been exposed to unauthorized parties (if necessary). This introduces a possibility that anyone can retrieve any piece of data from anyone, trusted or not, regardless of secure channel or not. The question is how can this idea be achieved? As Diana Smetters and Van Jacobsen says [SJ09], we must ensure the content’s validity, provenance and relevance.

- Validity - Complete and unmodified content from the publisher.
- Provenance - Should the publisher be trusted with the content requested?
- Relevance - Is the content what the requester intended?

There exist concepts to achieve these goals. One can do hash verification on the content to be sure that the content is unmodified. But there should also be a binding between the **Name** to the content. However, this does not provide provenance nor relevance because the publisher is not linked to the data, and thus we cannot be sure the publisher knows what the content contains. Hence there should be a linkage between the publisher, the **Name** and the content. A solution is to do a triple mapping of the **Name** (N) and content (C), cryptographically signed by the publisher (P) seen in Equation 2.1. This mapping is unique, relying on the hash computation done in the signing, providing validity, provenance and relevance. A requester can easily verify the **Name** and content binding, as well as authenticating that the data originates from the publisher who knows what the content is. Anybody can retrieve $M_{(N,P,C)}$ and verify the content and publisher mapping, hence an untrusted host and an insecure channel is not so bad anymore.

$$M_{(N,P,C)} = (N, C, \text{Sign}_P(N, C)) \quad (2.1)$$

A clear benefit of this approach is that it scales. The **Name** can be of any form because of the nature of hashing. Different naming rules should apply for different applications as there are no global naming rules that are optimal for each application.

This concept is integrated in the NDN protocol and it is required that every packet delivered from application layer is signed by the application. The protocol also provides an easy way for the application to encrypt data providing confidentiality. Encrypting the content with symmetric keys that are distributed to parties obtaining access right to the content together with the validity, provenance and relevance provides a way of securing data rather than securing communication channels.

Anonymity

Based on the nature of this architecture, NDN facilitates the practice of anonymity in the network. In a Tor network [DMS04], each node participating in a circuit only knows the two neighboring nodes. Only a global passive adversary that can monitor the entire network is able to decide the whole packet path, hence an adversary can know *who* is requesting and *who* is responding. Since the packet format (subsection 2.3.3) in NDN has no source or destination specific field as in a IP packet, the privacy of the network is more similar to a Tor network. If a packet

is captured at any arbitrary point of its path, the only information an adversary will get, is the two nodes between the packet capture and the content **Name**. Unless monitoring a complete network, it should be close to impossible to track packets. However, because of the semantic naming there are some issues related to privacy as it easily can be seen in the **Name** *what* the content contains in many cases. Also since signing of each **Interest** is required by the sender, some privacy information might leak. DiBenedetto et al. try to address these problems in [DGTU12] with an approach that use existing solutions from the Tor network. In 2010 the NDN-team planned to implement TORNADO [ZEB⁺10, Section 3.7], the NDN version of Tor, to demonstrate the privacy preservation capabilities of the network.

2.3.9 Attacks

Paolo Gasti et al. identifies several DoS attacks on NDN in their paper about DoS and Distributed Denial of Service (DDoS) in NDN [GTUZ13]. Other works have been done related to DoS in NDN [WCZ⁺14, SNO13, CCGT13]

In [LZZ⁺15] Zhang et al. propose an extension of the NDN protocol for addressing the access problem of cached **Data** in nodes. The NDN network is also potentially susceptible to content poisoning attacks which Ghali et al. addresses in [GTU14].

2.4 Related work

The work in this thesis builds upon three main concepts: synchronization, theoretical sensor networking and IBC. Some related work done will shortly be presented in this section.

There have been done work to show that NDN is well suited for synchronization. A synchronization application built by the NDN-team is ChronoSync [ZA13]. As explained in chapter 4, I use ChronoSync to achieve synchronization of files over NDN. There is also an application called iSync [FAC14], which is a scalable and high performance synchronization protocol.

Amadeo et al. [ACM14] propose a solution for reliable retrieval of **Data** from different wireless producers which can answer to the same **Interest** packet. This is highly applicable to a sensor network where you want to communicate with the closest sensor, e.g. the light in this room. In [ASLF14] Abid et al. simulate **Data** aggregation in wireless sensor networks. Jeff Burke et al. addresses efficient and secure sensing over NDN [BGNT14]. Burke has also contributed in developing and installing a system that secures building management systems at UCLA using NDN [SDM⁺14].

There is little research done on IBC in NDN. In [ZCX⁺11] Xinwen Zhang et al. propose a hybrid scheme with traditional PKI and IBC.

There is done some research on Elliptic Curve Cryptography (ECC) in Wireless Sensor Network (WSN) [LN08, SOS⁺08]. Leonardo B. Oliveira et al. [OAM⁺07] argued in 2007 that Identity-Based Encryption (IBE) is applicable to WSN. Oliveira et al. measured on a MICAz node (with the ATmega128 microcontroller, 8-bit/7.38 MHz processor, 4KB SRAM, 128KB flash memory) running TinyOS that the average execution time to compute a pairing is 30.21s. It is worth noticing that Yusnani Mohd Yussoff et al. [YHB12] tested an IBC implementation on ARM processor, and measured energy consumption on a device with a processor that runs at 20mA, 3.6V with frequency 667MHz to be 26.9 mJ.

Harsh Kupwade Patil and Stephen A. Szygenda have wrote a book about security for WSN using IBE [PS12]. The book explains relevant concepts in addition to summarize many research papers within the scope.

My contribution is unique due to

- the combination of using IBC in a WSN setting, running over NDN.
- the protocol proposals for *device registration* and *data pull* in a sensor network.

Chapter 3

Identity-Based Cryptography

This chapter will present the concept of IBE and Identity-Based Signature (IBS), and why it is highly applicable to use this type of cryptography in NDN. The possibilities to use the file synchronization module to do key distribution and revocation will be introduced.

3.1 Notations

Notations related to IBC used throughout this thesis is listed in Table 3.1.

Symbol	Description
MPK_i	Master Public Key belonging to i
MSK_i	Master Secret Key belonging to i
SK_i	Secret Key (private key) belonging to i
PK_i	Public Key belonging to i
ID_i	Identity belonging to i

Table 3.1: IBC notations used throughout the thesis.

3.2 Concept

IBE was first proposed by Shamir [Sha84] in 1984. Shamir proposed a scheme for IBS, but not a scheme for IBE. The concept of IBE builds upon every user having an Identity (ID) that is used as the Public Key (PK). This ID can be anything, i.e. email, phone number, Social Security Number (SSN), or a Name (subsection 2.3.4). The Secret Key (SK) that is extracted from the ID is issued by a TTP. Notice that if every user could have created their own SK, then so could anybody else with the same computational power, since the user does not obtain any “privileged” information about its ID [Bid06]. This eliminates the need of certificates because the

SK allocation itself is a verification by the TTP. The IBE implementation remained unsolved until 2001, when Dan Boneh and Matthew K. Franklin proposed [BF01]. However the scheme has only been shown to be secure within the random oracles model [Wat04], hence less practical.

IBE is based on performing asymmetric encryption with a publicly know ID working as the PK. As seen in Equation 3.1, the ID can be a `Name` (e.g. “/ndn/no/ntnu/haakon”). Hence the `Name` becomes the PK (from now referred to as ID). Therefore IBE is highly applicable to NDN.

$$ID_{device} = PK_{device} = Name_{device} \quad (3.1)$$

In IBE there is a TTP that is called Private Key Generator (PKG). The PKG's task is to extract a SK given an ID and provide public parameters (Master Public Key (MPK)) needed for performing encryption, decryption, signing and verifying. In Figure 3.1 the IBC methods is illustrated in practice. Equation 3.2 shows the key pair ID and SK which is used in IBC.

$$(ID_{device}, SK_{device}) \quad (3.2)$$

First the PKG runs `Setup()`. `device1` can then request a SK by sending `IDd1` to the PKG. In return the `device1` receives the `SKd1` as well as the `MPKPKG`. `device2`, which is already a part of the trust domain, sends a signed request for `Data` to `device1`. `device1` verifies the signature and responds to the request with a signed, encrypted content. `device0`, which do not have a SK generated from the PKG and thus is not a part of the trust domain, sends a request to `device1` that is declined.

1. `Setup()` generates the key pair (MPK, Master Secret Key (MSK)). These keys are used by only the PKG to extracting secret keys, encryption and decryption.
2. `Extract(MPKPKG, MSKPKG, IDdevice)` generates a secret key from a given ID.
3. `Encrypt(MPKPKG, IDdevice, message)` encrypts the message.
4. `Decrypt(MPKPKG, SKdevice, cipher)` decrypts the cipher generated from the encryption.
5. `Sign(MPKPKG, SKdevice, message)` signs a hash digest of the message (e.g. Secure Hash Algorithm 1 (SHA1)).
6. `Verify(MPKPKG, IDdevice, message, signature)` verifies the signature.

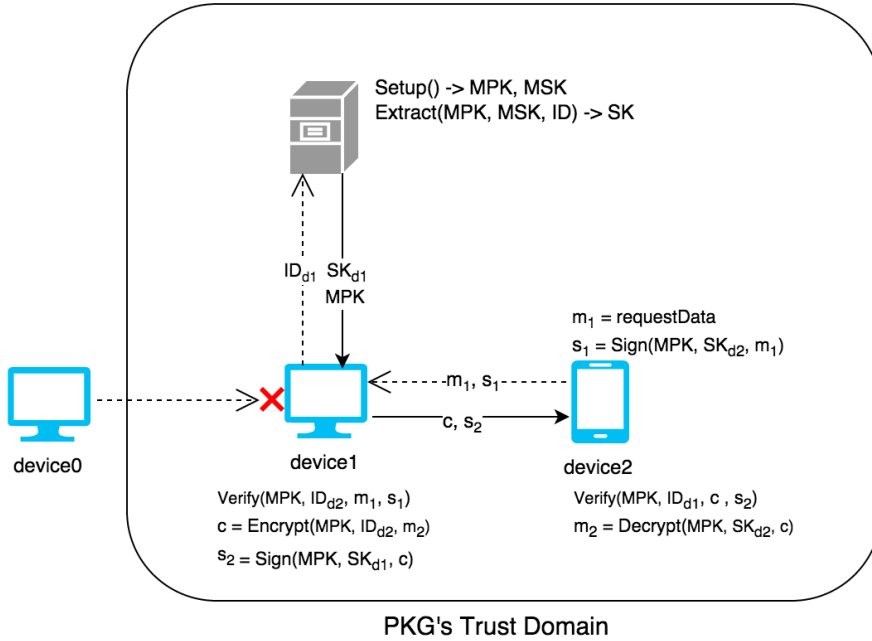


Figure 3.1: Methods of an IBC systems illustrated in practice. The PKG calls `Setup()` which generates the key pair (MPK, MSK) . Device1 then requests registration by sending its ID_{d1} to the PKG. PKG extracts the SK_{d1} composing the key pair (ID_{d1}, SK_{d1}) . After retrieving the SK_{d1} , device1 receives an request for a resource, signed by another device in the trust domain. The request is verified and the resource requested is encrypted and sent as a response. Finally, device2 verifies the response and decrypts the resource content. Device0 requests the same resource, but the request is denied because the verification fails, due to its inadequate signature.

To encrypt a message with IBE, the user encrypts a Content-Encryption Key (CEK) with the recipients ID. The user encrypts the message using the CEK together with symmetric encryption [AMS09, section 2.2.2], and sends both the encrypted CEK and the encrypted content to the requester.

It is two main concepts which holds a great part of the security in an IBC system. The security of IBC depends mainly on the secrecy of the PKG, therefore it is crucial to deploy a secure PKG. Also, it is important to identify each device before issuing SK. Approving wrong devices and allocating SK to an adversary would compromise the system.

There are some drawbacks related to IBE such as issues around trusting the PKG considering that the PKG generates all SKs. If the PKG is compromised by an

adversary, the adversary will retrieve all SKs belonging to the corresponding ID. Suspicion of Man In The Middle (MITM), where the PKG is the adversary, can be a problem for users. The same issue does however occur in Kerberos, which is a well recognized security system. Initializing might also be a problem because to allocate SKs, a secure channel has to be established. However, this is not a bugger problem than in existing networks. Pre-shared secrets or Diffie-Hellman key exchange might be a good solution.

3.3 Security

When designing protocols in cryptography one first usually designs an ideal system where all parties have random oracle access, then proves the security. A random oracle is like a “black box” that outputs truly random numbers. Second, one replaces the oracle access with a hash function. This gives an implementation of an ideal system in the real world, but without random oracles [BR93]. It is perfectly fine to make statements based on the ideal system, but debatable whether the same statements yields for the implementation in the real world. Canetti et al. concluded that there exist secure schemes in the *Random Oracle Model*, but for which any implementation of the random oracle results in insecure schemes [CGH04]. Boneh and Franklins IBE scheme is only secure when using random oracles, and relies on elliptic curves [BF01].

Following the *Standard Model* one does not resort to the random oracle heuristic and does not rely on non-standard complexity assumptions. Hence proving security in the standard model is preferably. In 2004 Boneh and Boyen proposed a fully secure scheme in the standard model [BB04]. However the scheme is not efficient.

The complexity assumptions is based on bilinear maps. Let \mathbb{G}_1 and \mathbb{G}_2 be groups of prime order p , and g be a generator of \mathbb{G} . We say that \mathbb{G} has a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ if e is efficiently computable, e is bilinear, i.e. $e(g^a, g^b) = e(g, g)^{ab}$ (for all a and b), and e is non-degenerate, i.e. $e(g, g) \neq 1$. For more details about bilinear maps and Bilinear Diffie-Hellman Problem (BDH) used in IBE, the reader is encouraged to take a look at [BF01, Nac05].

First practical scheme was introduces by Brent Waters [Wat04]. But as David Naccache states in his paper [Nac05], Waters’ scheme without random oracles introduces too large public parameters (164Kilobyte (KB)!). Naccache proves that he was able to construct a practical and fully secure scheme in the standard model based on the Decisional Bilinear Diffie-Hellman Problem (DBDH) assumption. The scheme is a modification of Waters’ scheme, but with public parameters of just a few KB size.

Waters created a fully secure IBE system with short parameters under simple assumption in 2009 [Wat09].

Chapter 4

Key Distribution Scheme

This chapter will first introduce the application that makes the key distribution scheme possible over NDN. The application, File Synchronization Module (FSM), is designed to be applicable to any other distribution of files over NDN. The key distribution scheme and key revocation I use in the chapter 5 will be presented.

4.1 File Synchronization Module

The FSM is built upon ChronoSync, which is explained in Appendix A. The goal for the FSM is to distribute **Data** to a large group of nodes. Each node wants to verify that the distributed **Data** originate from the publisher. Each node always want to obtain the newest version of the **Data**. One example where FSM is applicable is when we want to distribute a list of public keys within a domain. Let us say there is a list owner, e.g. a TTP that could be a university like the Norwegian University of Science and Technology (NTNU). NTNU wants to distribute to a large set of nodes, i.e. each student and employee at NTNU. Every node want to have every public key in NTNUs domain up-to-date. When the list of public keys gets updated, caused by for instance a key revocation or a key initialization, every node should immediately synchronize with the updated list.

There can be two types of roles in the FSM.

1. Distributor
2. Subscriber

Distributors are list-owners and have read-write access. Subscribers only have read access. A node can be both a distributor and a subscriber, and there can be several distributors that are equal, i.e. several owners of the list. However, there is one root distributor (i.e. the true owner) that should be able to delegate write

access to other nodes that should act as a distributor. One can verify the root distributor because the $ID_{\text{distributor}}$ is a part of the synchronization group's **Name** (i.e. `././<id_to_root_distributor>/./`). The capabilities is distributed to all nodes and signed by the root distributor. These capabilities is needed so that every node can verify the integrity and authenticity of the distributed list. If confidentiality is required, it can be achieved by symmetric encryption, and key exchange in the subscription protocol, i.e. using asymmetric encryption. However this becomes quite complicated concerning possible key leakage and redistribution of a new symmetric key when the number of subscribers is high. In the case of public keys list, the data would not have to be confidential, but rather rely on integrity and authenticity.

In Figure 4.1 the subscribers (**a**, **b** and **c**) wants to subscribe to the distributor's (**d**) list of public keys. In order to achieve this goal, the following actions should occur.

1. **d** announces that it wants to distribute a list to the network by registering the synchronization group prefix.
2. **a**, **b** and **c** ask for subscription to this list, and somehow authenticates themselves to **d** if confidentiality is required.
3. **d** approves those who should be approved, and returns a symmetric synchronization key. This step is only done if confidentiality is required.
4. **a**, **b** and **c** now knows that they are a part of the synchronization and have read access. They expresses a **Sync Interest** with their state, receiving **Sync Data** whenever **d** has announced a newer state.

4.2 Key Distribution

In traditional PKI, each public key is signed by a certificate authority and the generated certificate is sent as a response over a secure channel then validated by the the client. I want to make the certificate authority obsolete by distributing every ID with the PKG acting as a Key Distribution Center (KDC), as explained in the above section. In Figure 4.2 we see that the PKG multicasts the ID list to all devices that have joined the trust domain. Each device can verify the integrity and authenticity of the sync state **Data** and validate that the ID list surely originates from its own PKG, i.e. the distributor.

4.3 Key Revocation

Key revocation in systems are studied well in traditional PKI. However, few alternatives to revocation schemes in IBE PKI have been proposed. One suggestion is to

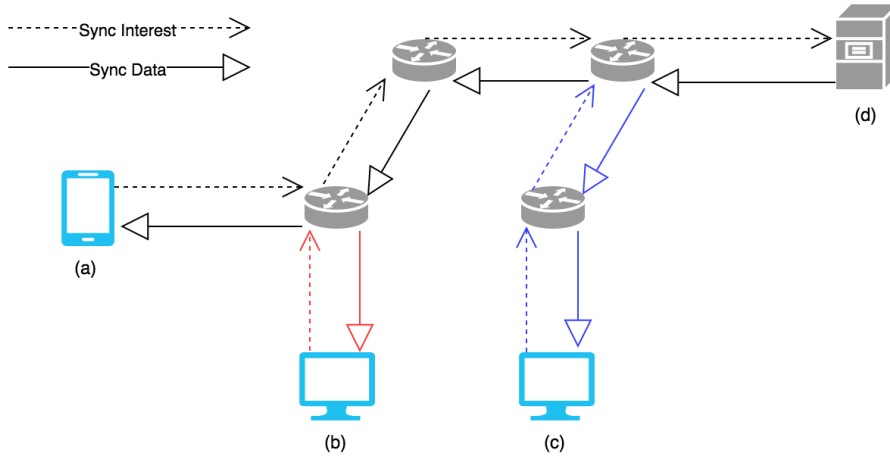


Figure 4.1: File Synchronization in NDN.

allocate secret keys with the ID combined with some sort of date, e.g. month-year or just year [BF01, section 1.1.1]. In this alternative a user has to renew its secret key each time the date changes, i.e. either the month or the year depending on the date format. The problem with this revocation solution is that it can be cumbersome for the PKG. Boldyreva et al. proposes a revocation scheme [BGK12] based on efficient key-update, which makes the workload for the PKG a lot easier. This scheme was only proven secure in the selective-ID setting where adversaries can attack an ID given they choose which one at the beginning of the game. The work done by Benoît Libert and Damien Vergnaud in [LV09] solves this problem. However, efficiently delegating both the key generation and revocation functionalities was a problem left open. Jae Hong Seo and Keita Emura solves this in [SE13].

Basically there exist two use cases where we want to renew a SK:

1. when the SK is compromised.
2. when the renewal period has expired.

If a key is compromised, we want to revoke the key immediately letting everybody know that this specific ID has been revoked. One problem with key revocation is that there is no way of revoking this key, and thus the ID has to be changed and distributed. However, a partially revocation can be sufficient in some networks. By partially, I do not mean revoking the SK, but rather only distributing the new ID to every node in the trust domain. In short, the compromised key should be

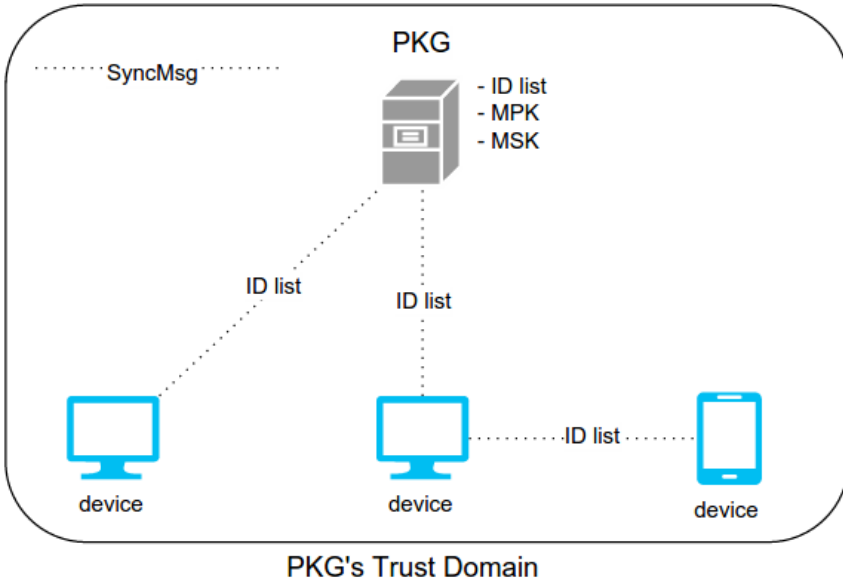


Figure 4.2: FSM with tree devices (subscribers) and a PKG (distributor).

removed from a distributed ID-list and the list containing only valid IDs should be disseminated. With the FSM, the list is distributed automatically when updated, and only a DoS attack together with a compromised key would make the system vulnerable.

Some periodic renewal might be necessary in many systems because it is not always known that a key has been compromised. If such an ID structure is applied, the key distribution should only contain the base key of each device.

Chapter 5

Sensor Application

In this chapter the sensor application will be presented. I present the threat model, the application protocols *device registration* and *data pull*, and finally an informal secure analysis of these protocols and the whole system. I also present a more formal analysis by modeling the protocols in Security Protocol Description Language (SPDL).

5.1 Health Sensors

There is an ongoing discussion of when the health technology revolution will come to human bodies now that IoT has become so popular. By revolution, I mean sensors placed in the human body. Sensors that can read your blood pressure, heart rate and measure insulin levels. Sensors that can detect whether your body is missing a substance, or if it is poisoned. There is no limit for what can be done. Everything that should be measured, will be measured by sensors integrated in the human body. But who will be able to read the data? Or perform instructions to the sensors/devices? There is some major privacy issues related to this discussion, and problems that needs to be solved.

In 2011, Jerome Radcliffe discovered that his insulin pump easily could be hacked [Rad]. Basically the pump would take instructions from anyone and do anything, with no questions asked. This is a worst case scenario when it comes to hacking medical devices attached to a human.

For this matter I propose a HSS that is built upon NDN with IBC ensuring a secure and locked environment. First, let me introduce you to The Stig. He has developed diabetes and he does not want to manually monitor his glucose levels and adjust the insulin pump every meal. He has injected a Continuous Glucose Monitor (CGM) to monitor his glucose levels and report to the insulin pump, automatically. In addition to his diabetes, he has a heart disease which forces him to monitor his heart rate at any given time. In Figure 5.1 we can see The Stig with all his sensors and devices.

The CGM reports periodically to the insulin pump, and all sensors reports to The Stig's mobile so that The Stig can watch what is going on.

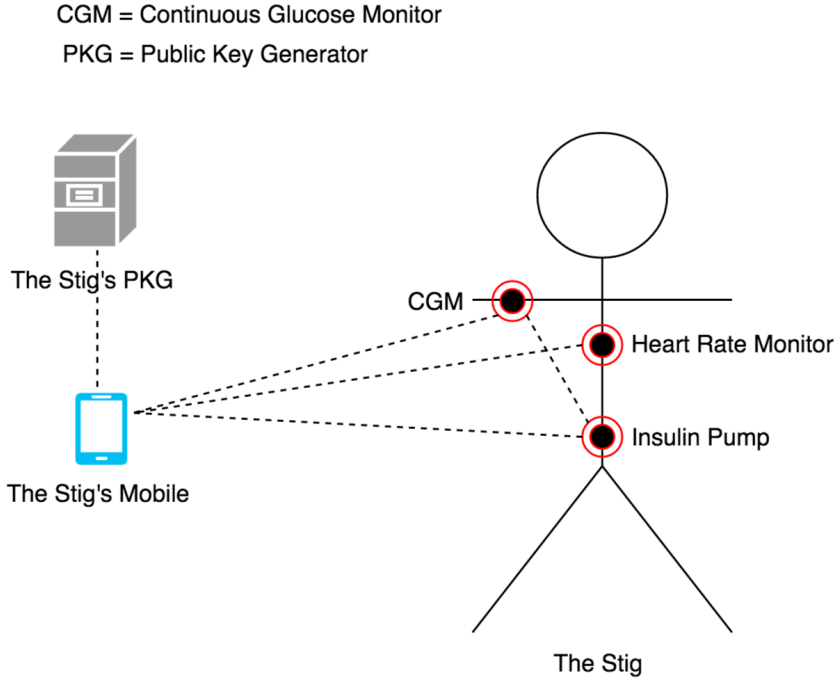


Figure 5.1: Health Sensor System (HSS). The image illustrates a trust domain administrated by the Stig. Devices exchange data securely and trust is established between them using the HSS trust model.

5.2 Health Sensor System

To obtain a secure system, trust have to be established between the sensors and the devices. There need to be integrity controls, confidentiality protection and access control. In the following sections, I will describe the protocols suggested for achieving the mentioned goals.

5.2.1 Rendezvous Authentication

One of the best solutions for authentication of an identity in cryptography is rendezvous authentication, the concept of meeting face-to-face for authenticating who you are talking to. Most cases in IoT we have the advantage of identifying devices in a physical manner. This means that it is possible to authenticate devices, such as

sensors. Typically, this kind of authentication will rely on 1) manually inspection and 2) digital connection, e.g. through Near Field Communication (NFC). In the proposed system, I assume that this type of authentication is achieved in a secure manner and do not discuss how this should be done. Typically this kind of authenticated connection should be used to preload a secret key to devices, establishing authentication in a later phase.

5.2.2 Threat Model

Threats that I find relevant for the HSS can be categorized into three main categories: threats to privacy, threats to availability and threats to control. I assume the following threat model:

1. An adversary might try to eavesdrop information (privacy).
2. An adversary might try to send bogus commands, e.g. injection, replay and MITM (control).
3. Jamming, node compromise (such as theft of mobile) and DoS (availability).

I assume that the PKG cannot be compromised by any adversary, and thus the MSK will always be hidden from any adversaries. However, it is extremely important that the machine which plays the role of the PKG is secured in a physical manner, as well as remote access security. The PKG is the single point of failure in the whole system.

An idea introduced by Aaditeshwar Seth and Srinivasan Keshav in [SK05, Section 5.4] is to avoid storing the SKs in devices that is more likely to be lost or stolen, e.g. a mobile. Using Hierarchical Identity-Based Cryptography (HIBC), one can extend the key hierarchy by another level that is time-based. These time-based keys can then be downloaded to the mobile on a daily basis, hence the time the mobile will be compromised is reduced.

5.2.3 System Initialization Phase

When The Stig is setting up his HSS, first he has to configure the PKG. Any type of computer can play the role of the PKG and The Stig has chosen his home server, from now “the PKG”. The Stig is now the only admin user in the system, hence he has full control over which device that should be granted access to the trust domain. The PKG runs the `Setup()` which creates key pairs that is used to do IBE and IBS. Second, he wants his mobile device, from now “the mobile”, to be a part of the PKGs trust domain, and further add all of the other devices and sensors, from now “device(s)”. This is done through a device registration phase.

5.2.4 Device Registration Phase

The goal for the device registration protocol is to achieve a secure one-round secret key exchange. For the protocol to be secure, there are several issues that need to be addressed. 1) The response message containing the secret key has to be encrypted. This can be achieved by using a pre-shared symmetric key to do encryption on the SK. 2) The response message has to be signed by the PKG for integrity and authenticity reasons. 3) A nonce has to be present for replay protection. The pre-shared symmetric key should be a temporary random generated key. Thus the device registration **Interest** between the device and the PKG can be unique. This implies that the device can authenticate itself in a way that an adversary cannot do.

The device registration is divided into two phases. Phase 1 is the sharing of a temporary random key \mathbf{tk} used to achieve confidentiality and authenticity in phase 2, illustrated in Figure 5.2. Phase 2 is the SK allocation, illustrated in Figure 5.3.

Phase 1. Trust is difficult to be established without a rendezvous authentication or any form of pre-established trust chain like e.g. a certificate chain. I do not assume that every device is a part of such trust chain before registration, and thus the trust between the device and the PKG has to be based on the concept explained in subsection 5.2.1, with manually inspection of the device and preloading of a temporary random key \mathbf{tk} . The packet flow of the key preloading, done over e.g. NFC, is shown in Figure 5.2. The device generates a temporary random key \mathbf{tk} and together with the ID_d , are loaded onto the PKG. The device receives the ID_{PKG} and the MPK_{PKG} in return. Since $ID_{PKG} = Name_{PKG}$, the device know the **Name** the **Init Interest** should have. It can also verify signatures from the PKG, with the MPK_{PKG} . Before phase 2, an admin user has to approve the requesting ID_d and the corresponding \mathbf{tk} .

Phase 2. Now that both the device and the PKG possesses the shared secret \mathbf{tk} and the device has been approved by an admin user, phase 2 can begin. The device sends an **Init Interest** that contains its ID_d and a nonce which is encrypted with the \mathbf{tk} . The PKG decrypts the message and uses the received ID_d to extract the SK for the device (this will be the SK_d belonging to the PKGs trust domain) and uses the \mathbf{tk} to do symmetric Advanced Encryption Standard (AES) encryption on the secret key. The **Data** response to the **Init Interest** will contain the encrypted SK_d , the nonce and a signature. To finish the device registration protocol, the device decrypts the SK_d , checks the nonce and verify that the SK allocated actually belongs to the earlier received MPK and that is corresponds to its ID_d . The device has established a trust with its PKG and can verify other devices within this trust domain.

Now that the mobile is authenticated, devices can connect to the mobile through e.g. NFC for device registration. This results in a rendezvous authentication between the

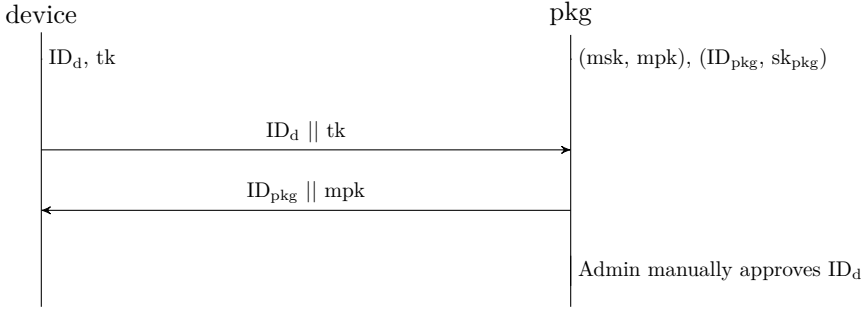


Figure 5.2: Device Registration, phase 1. The messages are exchanged through e.g. NFC (subsection 5.2.1), and thus are protected against any adversaries outside the range of the NFC signal (~ 1 meter). At first the Device generates a temporary random key \mathbf{tk} and sends this key together with its \mathbf{ID}_d to the PKG. The PKG responds with its \mathbf{ID}_{PKG} and $\mathbf{MPK}_{\text{PKG}}$. Finally an admin user has to approve the requesting device.

device and the mobile, and if the mobile is given the authorities to perform device registration (Access Control List (ACL) in subsection 5.3.1), the new device can join the PKGs trust domain.

Security Analysis

It is important that the protocol possesses properties such as privacy, availability and control. I shortly present a formal security analysis by modeling the protocol in SPDL and verifying certain claims through Scyther [Cre08]. After that, an informal discussion of the security in the protocol will be presented.

Scyther. The analysis proves that the protocol is confidential, replay and injection resistant, and possesses properties such as integrity and authenticity. All claims made (i.e. `alive`, `secret`, `weakagree`, `niagree`, `nisynch`) holds, thus no attacks is detected in Scyther. The SPDL code of the protocol can be reviewed in section B.1.

Authenticity. The protocol holds the required authenticity and integrity. The pre-shared temporary random key \mathbf{tk} is shared in an assumed secure manner, thus appending an encrypted message to the `Init Interest` will lead to authentication for the PKG as it will decrypt the cipher with \mathbf{tk} authenticating the device. Thus the encryption, with the \mathbf{tk} as key, protects against MITM attacks. The response `Data` is hashed and signed with the PKGs SK which provides authenticity and integrity for the requesting device since it has already retrieved the MPK and \mathbf{ID}_{PKG} in phase 1. The signature can easily be verified, and since an adversary do not obtain a polynomial time algorithm that can forge the SK the device can be sure that the

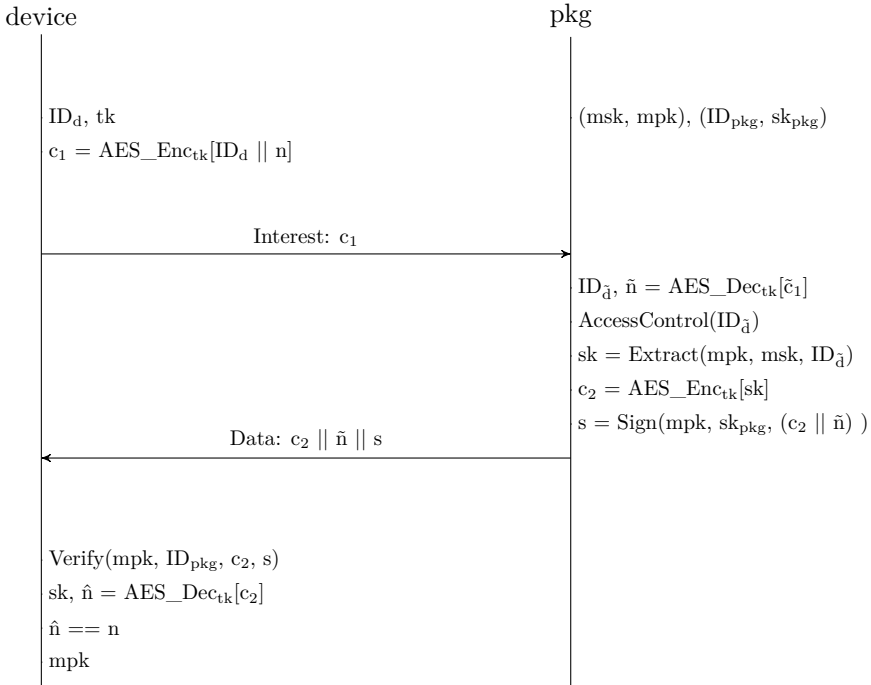


Figure 5.3: Device Registration, phase 2. The device sends a **Init Interest** encrypting nonce n and its ID_d . The device receives the response, decrypts the cipher c_1 , checks if the ID_d is approved, extracts the secret key corresponding to ID_d , encrypts the secret key and finally signs the **Data**. The device verifies the nonce and decrypts the received cipher, obtaining the SK_d .

message is signed by the corresponding ID_{PKG} . Thus the signature protects against MITM attacks.

Confidentiality. The SK will be encrypted with the pre-shared temporary random key tk , and thus the confidentiality is preserved. An adversary will only be able to know the MPK, cipher texts, signatures and both IDs, which is not required to be confidential and not sufficient to compute the SK that is extracted. The adversary do not obtain a polynomial time algorithm that can forge the signature or the tk from the known parameters. Hence an adversary has to obtain an algorithm to compute the SKs, which is the same polynomial time algorithm as in the sub section above that the adversary do not have access to.

Replay. Since the adversary cannot compute tk nor forge the signature, it cannot send encrypted, signed **Interest** nor **Data** that is captured at any arbitrary point. Devices keep track of the nonce corresponding to a device registration, hence a replay

with will be detected and thrown away.

5.2.5 Deployment Phase

The goal for the data pull protocol is to achieve a secure one-round data pull with authorization and integrity. For the protocol to work and the data pull to be successful, 1) both devices has to belong to the same trust domain (i.e. has registered with the same PKG) and 2) the requester has to have granted access rights for the resource requested.

The flow of the protocol is illustrated in Figure 5.4. First the requester has to express an **Interest** to the target device asking for a specific resource. The requester signs the **Interest** and appends it to the content **Name**. The publisher checks whether the requester has access rights to the requested resource and verifies that the requester is a part of the same trust domain. If the requester is authorized, the publisher responds with the **Data** containing the resource. The publisher will also do a symmetric encryption on the **Sensor Data** and do a asymmetric encryption on the CEK with the requester's ID. This step is only performed if **Data** confidentiality is needed. Then the **Data** packet is signed and sent. Finally the requester receives the **Data**, verifies the signature and decrypts the **Sensor Data**.

Security Analysis

It is important that the protocol possesses properties such as privacy, availability and control. I shortly present a formal security analysis by modeling the protocol in SPDL and verifying certain claims through Scyther. After that, an informal discussion of the security in the protocol will be presented.

Scyther. The analysis proves that the protocol is confidential, replay and injection resistant, and possesses properties such as integrity and authenticity. All claims made (i.e. **alive**, **secret**, **weakagree**, **niagree**, **nisynch**) shows no attacks in Scyther. The SPDL code of the protocol can be reviewed in section B.2.

Authenticity. The protocol holds the required authenticity and integrity. The message is hashed and signed with the senders SK which provides authenticity and integrity. The signature can easily be verified by the receiver. Since an adversary do not obtain a polynomial time algorithm that can forge the SK one can be sure that the message is signed by the corresponding ID. Thus the signature protects against MITM attacks.

Confidentiality. All **Data** that flows through the HSS can be encrypted if necessary. When a resource is requested, the publisher will do an access control to decide whether the $ID_{\text{requester}}$ has the right capabilities. The CEK will be asymmetrically encrypted, and the resource data will be symmetrically encrypted, thus both key and

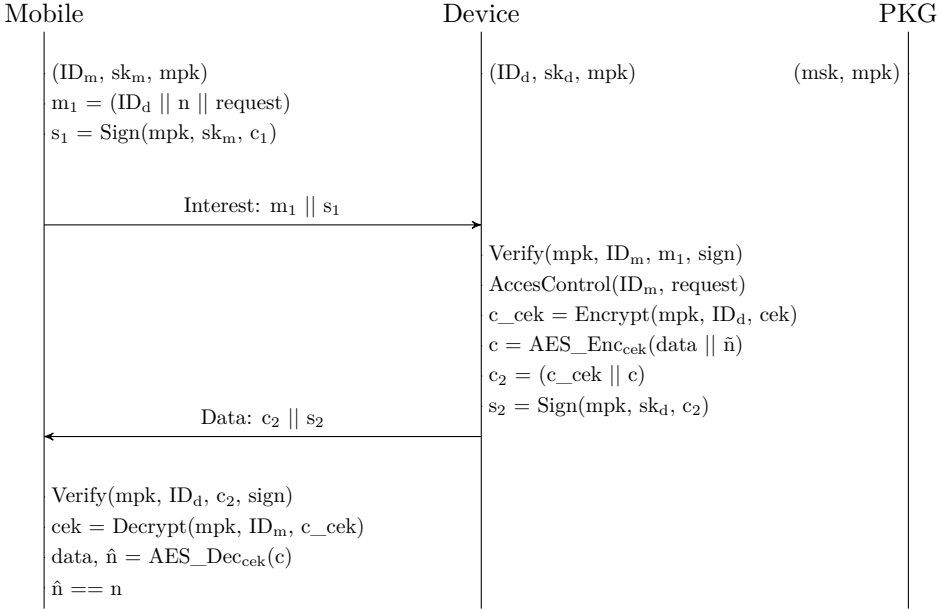


Figure 5.4: Data pull under deployment. The mobile sends a Sensor Interest to the device appending the request for a resource and a nonce n . The Interest is signed with the mobile's secret key sk_m and verified by the device. The device checks whether the mobile has a valid capability for the requested resource and encrypts the data if granted. The Data response is signed with the device's secret key sk_d . The mobile decrypts of the content encryption key cek and the cipher c , checks whether the received nonce \hat{n} is equal to n and finally accepts the data as correct.

data is confidential and only available to whoever has the corresponding $SK_{requester}$ to the $ID_{requester}$. An adversary will only be able to know the MPK, nonce, request, cipher and both IDs, which is not required to be confidential and not sufficient to compute the resource data. The adversary do not obtain a polynomial time algorithm that can compute the resource data from the cipher and other known parameters. Hence an adversary has to obtain a algorithm to compute the SKs, which is the same polynomial time algorithm as in the sub section above that the adversary do not have access to.

Replay. Since the adversary cannot forge the signature, it cannot send **Interest** nor **Data** that is captured at any arbitrary point. This is due to the nonce presence in all packets. Devices keep track of the nonce corresponding to a data pull, hence a replay with will be detected and thrown away.

5.2.6 Key Distribution using File Synchronization Module

The Stig wants to have full control over the devices that are a part of the trust domain, and be able to remove a device if necessary. Each device should have an updated list of all IDs. The distribution of this list can easily be achieved by using the FSM (section 4.1 & section 4.2). The PKG will be the distributor in this synchronization and each device will be a subscriber.

5.3 Informal Security Analysis

In this section an informal security analysis of the whole system is presented. Access control will first be presented, along with CIA, and finally the trust model.

5.3.1 Access Control

Since the ID_{device} is appended to the **Interest** and the **Interest** is signed by the corresponding SK_{device} , the ID of the device can easily be authenticated. When a device retrieves an **Interest** for its **Sensor Data**, there should be an authorization mechanism on the requested resource. One can argue that once a device has been authenticated in the PKG's trust domain, everyone in the domain can be sure that each device will not abuse the resources available. However, due to scalability and cautiousness this is not a secure way to handle access control. If a device does not need a privilege, it does not need it. Hence it should not have it. That is the least privilege access principle, which is default in Capability Based Approach to IoT Access Control [GPR12]. This approach has some additional benefits for the HSS, such as

- delegation support - A device can grant access rights to other devices, as well as granting the right to further delegate these rights to a third device.
- capability revocation - If the PKG has granted delegation rights to a mobile, and the mobile is not found trustworthy after some time, the capabilities issued by the mobile can easily be revoked.
- information granularity - Specific resources from a device can be granted access to in different granularity.

Another solution can be an ACL based approach equivalent to what Wentao Shang et al. did in [SDM⁺14].

5.3.2 Confidentiality

Generally, the confidentiality is achieved by doing asymmetric encryption on a CEK that is used for symmetric encryption on the content. As explained in the sequence diagram (Figure 5.4) presented in the above sections, each **Interest** appends the requester's ID (Equation 3.1). Since the $ID_{\text{requester}}$ always is appended it can always be used to do asymmetric encryption, hence all CEKs can be encrypted only for the requesting device, and thus the confidentiality in the system can always be achieved.

5.3.3 Integrity and Authenticity

Each device will obtain a SK allocated by its superior PKG, as explained in section 3.2. With the concept from rendezvous authentication (subsection 5.2.1) together with the PKGs MPK, one can trust that the device is authorized to the PKGs trust domain. Hence all signed packets can be verified by anyone with the MPK. In this setting, a verified signature acts as an assurance of authentication and integrity.

5.3.4 Availability

This is a harder problem to solve. The network is purely wireless, hence vulnerable to jamming.

An adversary could try to send infinite **Interests** to a device with an invalid signature, hence the device may be overloaded with work and might run out of battery fast. Therefore one should check the MPK before doing any crypto. This is also why I have chosen to append the MPK in the packet, illustrated in Figure 6.3.

5.3.5 Trust Model

For a system to be secure, cryptography together with trust is essential. The HSS trust model is built upon trusting a centralized authority (typically the user's home server), rendezvous authentication and IBC. The fact that it runs over NDN makes it easier to achieve security goals and usability for developers.

The ID is the **Name** of the device, and all content published by the device will begin with this **Name**, hence it is easy to verify that the publisher is the owner of the content. To be able to verify and encrypt messages, the device need the MPK_{PKG} and the ID of the user it wants to communicate with. If allowed by the PKG, the public parameters MPK_{PKG} is public for everybody that is not a part of the trust domain as well. To be able to sign and decrypt messages the device has to be verified by an administrator that controls the PKG and be issued a SK such that $(ID_{\text{device}}, SK_{\text{device}})$. Because every device trust its administrator and likewise is trusted by its administrator, each device should trust other devices after verifying their signature.

As stated, it is assumed that an adversary do not obtain a polynomial time algorithm that can forge a valid signature or forge a valid SK.

As proposed, the application should follow the least privilege access principle.

Chapter 6 Implementation and Testing

This chapter will first introduce the most significant frameworks that must be installed to be able to run NDN applications. Then the design and architecture will be explained. The source code is left out from the appendix due to the size inconvenience, but referenced as an open GitHub repository [Mø15]. Finally, the test result will be presented.

6.1 Installing Named Data Networking Forwarding Daemon

Several libraries is required for experimenting in a NDN environment. Installation guides can be found at the Github project [NT15a]. First we need to install the Named Data Networking C++ library with eXperimental eXtension (ndn-cxx). ndn-cxx is a implementation of NDN primitives. It is a fundamental framework that NDN application requires. Second we need to install the NFD [AMY⁺] which is a network forwarder and also in the core implementation of NDN. The major modules implemented in NFD is:

- Core - Common services shared between the different NFD modules (such as hash, DNS resolver, face monitoring etc.).
- Faces - Generalization of different interfaces, explained in subsection 2.3.5.
- Tables - PIT, CS, FIB, explained in subsection 2.3.5.
- Forwarding - Packet processing.
- Management - Enables users/programs to interact with the NFD forwarder state.
- Routing Information Base (RIB) Management - Managing routing protocols and application prefix registration.

The NDN project is under development, and thus the implementation of NFD has its deficiencies. Ideally we want the devices to communicate directly with each other using WiFi, without running over IP. This face functionality is not yet implemented, and thus NDN is running over IP in my experiments.

6.2 Installing PyNDN2

The work done in this thesis is written in Python, hence the PyNDN2 [NT15b] is used. This is an easy to use implementation of NDN and comes with great code examples.

Because the NDN protocol require signing of `Data` packets (subsection 2.3.8) some new implementation in the PyNDN2 source code was necessary to be able to sign and verify with IBS. I added the `python/pyndn/sha256_with_ibswaters_signature.py` file that follows the pattern of the existing RSA Signature (`python/pyndn/sha256_with_rsa_signature.py`) and is of type `Signature`. Some small additions in the `python/pyndn/encoding/tlv_0_1_1_wire_format.py` and the `python/pyndn/encoding/tlv/tlv.py` is added so PyNDN2 recognizes the IBS when the `Data` packet is encoded and decoded. The files edited can be found under `src/other/charm/` in [Mø15].

6.3 Installing Identity-Based Cryptography

To be able to run IBC the Pairing-Based Cryptosystems (PBC) [Ben07] needs to be installed. I use the Charm framework [AAG⁺13] which implements several IBE and IBS schemes in Python. Charm is a framework for rapidly prototyping cryptosystems.

Some small modifications had to be done in the Waters-IBS [Wat04] implementation in Charm. In `charm/schemes/pksig/pksig_waters.py` there is a global variable, i.e. `waters`, that is used throughout all the methods in `pksig_waters.py`. The problem is that this variable is declared in the `setup()`, which is only called at PKG, and not by another devices that do not play the role of a PKG. And thus, the declaration of `waters` must be moved to the `__init__()` in `pksig_waters.py`, which is called by every device using the scheme. The file can be found under `src/other/pyndn/` in [Mø15].

6.4 File Synchronization Module - Implementation

FSM is a python application that runs over NDN and synchronizes all files in a specified path, with all participants within the synchronization room. Application goals are explained in section 4.1. The module is highly based on the Python implementation of ChronoSync [NT15b, test-chrono-chat.py]. The code can be retrieved from the thesis work repository [Mø15, fileSync.py]

The implementation of the FSM does not correspond 100% to the application goals. The distributor/subscriber model is not implemented, and neither is the IBS. The latter is because all packets that are sent is managed by ChronoSync. ChronoSync uses the PyNDN2 KeyChain to sign and verify all **Interest** and **Data** packets. The IBS integration explained in section 6.2 is not implemented with the KeyChain, hence this is added to the future work. I do however demonstrate that it works perfectly fine to perform both IBE and IBS over NDN in the HSS implementation.

The module triggers synchronization when files that are watched is changed, or when a file is added or removed. A library that makes it possible to watch files in OS X, Linux or Windows, is Watchdog [Pyt]. The implementation is illustrated in the class FileWatch in Figure 6.1.

6.5 Health Sensor System - Implementation

The HSS is a python application that runs over NDN. Application flow is explained in chapter 5. The implementation does not deal with sensor data retrieval from actual sensors, nor deal with sending instructions from devices to each other, but rather focuses on the trust model and security in protocols between devices in the network. The code is divided into several pieces shown in Figure 6.1.

The Device class [Mø15, device.py] implements the role of a device that can express **Interest** and offer **Data**. The PKG class [Mø15, publicKeyGenerator.py] implements the role of a PKG. Both classes uses PyNDN2, to be able to run over NDN, and IndentityBasedCryptography, to be able to perform IBC. IdentityBasedCrypto [Mø15, identityBasedCrypto.py] implements two IBE schemes and one IBS scheme from Charm. These schemes are listed below:

Waters05 [Nac05] that is a variant of Brent Waters IBE scheme [Wat04], but with smaller key size, hence more practical.

Waters09 [Wat09] that is also a fully secure implementation of IBE scheme.

Waters [Wat04] that is a implementation of IBS scheme.

6.5.1 Key Storage

Storing of secret keys should be done in a secure fashion. Sufficient key storage is not implemented.

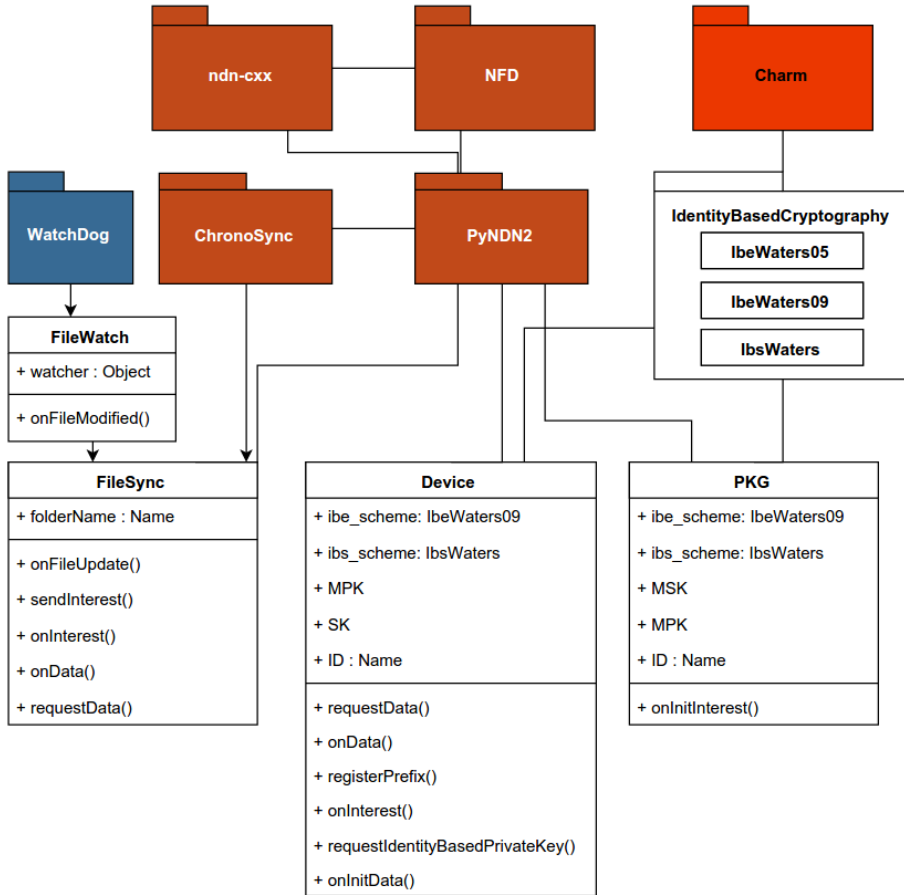


Figure 6.1: Package and class topology related to the work done in conjunction to this thesis.

6.5.2 Name Structure

As explained in the Name subsection 2.3.4, the Name structure in NDN is left to application developers to decide. Using IBC, it is essential that the ID is publicly known. Therefore the ID of every device should be concatenated with the Name. A device register the prefix `/ndn/no/ntnu/<device>/<resource>` and hence its ID is `/ndn/no/ntnu/<device>`.

6.5.3 Access Control

In access control subsection 5.3.1 I present a possible solution for access control. This is however not implemented in the application, because it is considered too high workload for this thesis, the suggested solution is implemented in Java, and not Python.

6.5.4 Packet Design

The packet format is designed with Google Protocol Buffers, which is a language-neutral, platform-neutral, extensible mechanism for serializing structured data.¹ Device registration packets have the structure presented in Figure 6.2. Initially, the idea was to have the cipher appended to the content **Name**. However, I experienced a problem where the **Init Data** never arrived at destination node. After some research in `ndn-cxx` documentation I found that the packets have a `MAX_NDN_PACKET_SIZE` of 8800 bytes and the **Init Data** exceeded this limit and reached 8904 bytes. Because the cipher is approximately 2KB and was appended to the content **Name** in the **Interest**, the **Data** response of course had to have the same content **Name**, hence 2KB overhead in the **Name**. The cipher can as easily be appended to the **KeyLocator Name**, hence the **Data** response can be 2KB less, resulting to a 6866 bytes **Init Data** packet.

Sensor packets have the structure presented in Figure 6.3. The code can be reviewed in [Mø15, messageBuf.proto].

Init Interest. The **Init Interest** can be seen in Figure 6.2 and consist of three fields: Content Name, KeyLocator and MustBeFresh. KeyLocator can be of type **Name**. As described in the NDN Packet Format [NT], generally this field can be used to specify where to download the certificate used to sign the **Interest**. However, in the trust model I use this field to publish the requesters **Name**, i.e. the requesters public key. This is very useful when using IBE and IBS.

Init Data. The **Data** response to the **Init Interest** is illustrated in Figure 6.2.

Sensor Interest. As in the **Init Interest** the KeyLocator field is used to define the $ID_{\text{requester}}$. The packet is illustrated in Figure 6.3.

Sensor Data. The **Data** response to the **Sensor Interest** uses the same structure as the **Init Data**. It is illustrated in Figure 6.3.

The **Init** and **Sensor Data** responses in the HSS have a structure that is defined in [Mø15, messageBuf.proto]. The fields are:

¹Google Protocol Buffers - <https://developers.google.com/protocol-buffers/>

- `MessageType` is an `enum` and can be either `Init` or `Sensor`.
- `EncAlgorithm` is an `enum` and represents which type of encryption scheme is used on the content.
- `IbeAlgorithm` is an `enum` and represents which type of IBE scheme is used on the CEK.
- `IbsAlgorithm` is an `enum` and represents which type of IBS scheme is used to sign the `Data`.
- `MasterPublicKey` is the PKGs public parameters used to do IBE.
- `SignatureMasterPublicKey` is the PKGs public parameters used to do IBS.
- `SymmetricKey` is the symmetric key used to encrypt the content. The key is encrypted. Only used in `Sensor Data`.
- `Cipher` is the encrypted content.
- `Session` is a nonce.

Init Interest

Content Name	KeyLocator	MustBeFresh
/ndn/no/ntnu/<target>/initDevice/	Cipher: <ID, nonce>	True

Init Data

Content Name	Signature	Data
/ndn/no/ntnu/<target>/initDevice/<session>	<signature>	{.....}

MessageType	EncAlgorithm	IbeAlgorithm	IbsAlgorithm	MasterPublicKey	SignatureMasterPublicKey
INIT	AES	WATERS09	WATERS	<mpk>	<smpk>
Cipher: <encrypted_secret_key>					Session: <nonce>

Figure 6.2: Init Interest and Data

6.5.5 Running the Code

First the NFD must be started on each device shown in Listing 6.1, if not already running. Then we have to make sure that each device participating in the network know the `Name` and IP address binding, since the testing will run NDN over IP. This

Sensor Interest

Content Name	Signature	KeyLocator	MustBeFresh
/ndn/no/ntnu/<target>/sensorPull/<session>	<signature>	/ndn/no/ntnu/<source>	True

Sensor Data

Content Name	Signature	Data
/ndn/no/ntnu/<target>/sensorPull/<session>	<signature>	{.....}

MessageType	EncAlgorithm	IbeAlgorithm	IbsAlgorithm	MasterPublicKey	SignatureMasterPublicKey	SymmetricKey
SENSOR_DATA	AES	WATERS09	WATERS	<mpk>	<smpk>	<cek>
Cipher: <encrypted_data>					Session: <nonce>	

Figure 6.3: Sensor Interest and Data

is accomplished by registering the mapping in the FIB at each device showed in the second line in Listing 6.1.

On the device playing the role of the PKG, run the code presented in Listing 6.2. This will create the key pair MPK_{pkg} and MSK_{pkg} and register the prefix where the other nodes can find the PKG.

On the device playing the role of e.g. a sensor, run the code presented in Listing 6.3. This will automatically register the prefix of the sensor, and start the initialize protocol with the PKG.

On the device playing the role of the user device (e.g. a mobile), run the code presented in Listing 6.4. This will automatically start the initialize protocol with the PKG. Running `r` will make the device expressing an **Interest** for sensor **Data** from the sensor.

```

1  $ nfd-start
2  $ nfdc register /ndn/no/ntnu/<data-device> udp://<device-ip-address>
3  $ nfdc register /ndn/no/ntnu/<pull-device> udp://<device-ip-address>
4  $ nfdc register /ndn/no/ntnu/<pkg> udp://<pkg-ip-address>

```

Listing 6.1: Start NFD

```

1  $ python application.py
2  $ pkg

```

Listing 6.2: Start PKG

```

1  $ python application.py

```

```
2 $ data
```

Listing 6.3: Start a device registering a prefix.

```
1 $ python application.py
2 $ pull
3 $ r
```

Listing 6.4: Start a device that will express `Interest` in `Data`.

6.6 Testing

In this section it will be presented which computers will be used during testing. The testing results will be presented together with the key/content sizes related to the HSS.

6.6.1 Computers

The plan was to test the application with several Raspberry Pi's to simulate a sensor network, with limited computation power. However this is not possible with the Charm framework as it is not compatible with ARM processors. The HSS is tested over several computers presented in Table 6.1. Each computer is assigned an ID which will be used for reference in the performance measurements.

ID	Computer	Operating System	Processor
C1	Macbook Pro	64-bit OS X 10.10	Intel Core i7 @ 2.0GHz
C2	Garsbook	64-bit Ubuntu 14.04 LTS	Intel Core i5 @ 3.0GHz
C3	HP	64-bit Ubuntu 14.04 LTS	Intel Core i7 @ 2.8GHz

Table 6.1: Computers used during tests.

6.6.2 Key Sizes

It is listed in Table 6.2 the different sizes for keys related to the IBE and IBS that is used in the HSS implementation. The CEK is a random \mathbb{G}_T element (section 3.3), and extracted to 40 bytes when performing encryption and decryption with AES. I would prefer to extract and send the extracted version of the CEK, i.e. the hash value of 40 bytes, but the implementation of the IBE encryption scheme demands a certain type of format for the input, and thus the whole CEK must be sent.

6.6.3 Performance

To be able to evaluate if IBC is applicable to devices with small computation power and limited battery, it has to at least perform somewhat in the range of regular

Data	Scheme	Size
Content Encryption Key (CEK)	Hash(G_T)	244 bytes
IBE Master Public Key	Waters09	2014 bytes
IBE Secret Key (SK)	Waters09	1164 bytes
IBE Encrypted CEK	Waters09	1472 bytes
Encrypted SK	AES	1633 bytes
IBS Master Public Key	Waters	2360 bytes
IBS Secret Key (SSK)	Waters	260 bytes
IBS Signature	Waters	412 bytes
Encrypted SSK	AES	437 bytes

Table 6.2: Sizes of different keys used in the health sensor system implementation.

asymmetric encryption (read RSA), and signing. Naccache suggested that if the prime p is 1024-bit, the scheme would provide equivalent security as a RSA 1024-bit key. For comparison reasons, the RSA key pair is therefore generated with the size of 1024-bit. In Table 6.3 the performance from running different cryptographic methods on the computers listed in Table 6.1 are presented.

Method	Scheme	C1	C2	C3
IBE PKG key pair generation	Waters09	99.65 ms	27.09 ms	36.08 ms
IBE Secret Key (SK) generation	Waters09	56.14 ms	17.86 ms	23.27 ms
IBE Encrypting CEK	Waters09	41.65 ms	18.91 ms	24.86 ms
IBE Decrypting CEK	Waters09	20.70 ms	9.87 ms	12.86 ms
Encrypting SK	AES	0.13 ms	0.10 ms	0.15 ms
IBS PKG key pair generation	Waters	97.55 ms	27.15 ms	35.02 ms
IBS Secret Key (SSK) generation	Waters	9.76 ms	2.87 ms	3.72 ms
IBS Sign	Waters	9.90 ms	2.88 ms	3.69 ms
IBS Verify	Waters	7.58 ms	2.66 ms	4.32 ms
Encrypting SSK	AES	0.06 ms	0.02 ms	0.04 ms
RSA (1024-bit) key pair generation	RSA	254.27 ms	119.34 ms	165.99 ms
RSA Encryption (40 bytes)	RSA	14.80 ms	4.40 ms	6.52 ms
RSA Decryption (40 bytes)	RSA	14.72 ms	4.45 ms	6.49 ms
RSA Sign	RSA	16.15 ms	4.59 ms	6.69 ms
RSA Verify	RSA	15.72 ms	4.53 ms	6.74 ms

Table 6.3: Cryptographic methods time chart. Each measurement is the mean time of 100 rounds and measured in milliseconds.

The device registration protocol described in subsection 5.2.4 and the data pull

protocol described in subsection 5.2.5 is tested on the computers listed in Table 6.1. The results of the round trip time are presented in Table 6.4.

Protocol	C1	C2	C3
Device Registration	57.7 ms	23.8 ms	32.3 ms
Data Pull	61.4 ms	31.0 ms	46.2 ms

Table 6.4: Round trip time chart. Time is measured in milliseconds.

The HSS is tested on two of the computers in Table 6.1. The topology is shown in Figure 6.4.

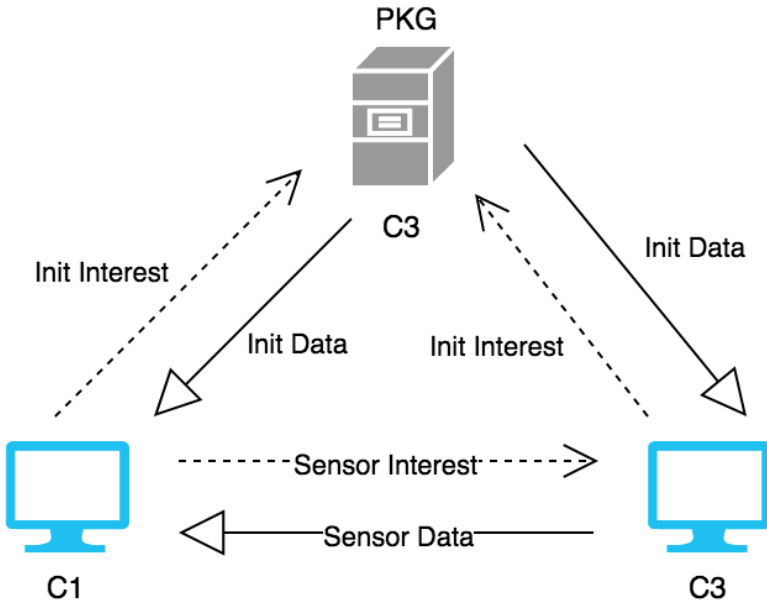


Figure 6.4: Health Sensor System implementation tested over two computer. C3 runs two nodes, i.e. the PKG and one device. C1 runs a second device.

6.7 NDN Testbed

The NDN testbed is a network of NDN nodes created for research purpose. NTNU joined the testbed contributing with the 24th node in the NDN testbed. The map of the NDN testbed is shown in Figure 6.5. The NTNU node is used during testing and experimentation.

Chapter 7

Discussion

In this chapter the work done in conjunction to this thesis will be discussed. First I will talk about NDN and the pros and cons using IBC in NDN. Then I will discuss some central topics in the HSS. Testing results, scalability issues and other applicable networks for the application will be mentioned.

7.1 Named Data Networking

NDN facilitates a lot of concepts that shows to be a huge benefit for today's Internet, and the predicted increase of IoT. The naming of content and content routing provides usability to IoT and WSN. Bandwidth redundancy in the network is reduced, security properties in network layer is provided, and the linkage between data and its publisher can easily be proven. It is easier for machines to communicate directly, without having to interconnect through a router. Broadcast and multicast comes naturally, hence wireless communication can be done in a simple manner.

Developing applications on top of NDN is easy once a basic perception of the NDN architecture is understood. The PyNDN2 framework comes with good examples of how to develop simple applications with packets that are signed and encrypted.

The concept of naming data introduces more simplicity, but also a new way of application design thinking. Addressing and security is dealt with in one place in the architecture compared to in an equivalent system over IP. A problem with WSN in IP networks, is that it is a limited number of IP addresses (especially in IPv4). So the global scalability issue arises due to the potentially large number of sensors that could be deployed. With the naming rules in NDN, this is not an issue. Security is easily applied in NDN which is shown to be a huge problem to many systems nowadays.

A huge advantage is that one can ask the network for content, and easily verify the signature. The **Interest** a requester expresses, has the same content **Name** as

the **Data** received in return. Hence the signature will be signed by a $SK_{\text{publisher}}$ corresponding to the ID part of the content **Name**.

7.2 Identity-Based Cryptography in Named Data Networking

The concept of IBC appears to be highly applicable to IoT and WSN [PS12]. Running IBC over NDN, makes it even more practical, because of the naming concept that NDN is built upon. As mentioned it is easier to secure data, relate data to publisher, and authenticate that the publisher is aware of what content it published. I believe that using IBC in a WSN running over NDN should make applications with security less complex and more practical than using security such as RSA running over IP.

Using ID as public key eliminates the binding of ID and certificate. Compared to ordinary PKI where the recipient have to download the public key certificate to verify the digital signature. This is practical and results in less communication overhead when establishing connection, and reduces energy consumption. IBC implies less keys involved, only SK and MPK have to be stored at each device. IDs of each device have to be known and distributed anyway in existing sensor networks (IP addresses). Also, the mapping done by DNS is eliminated, because IP addresses is no longer needed. Exchanging data between nodes can be done with cryptography completely without the PKG after device registration. However, there is an issue of having a TTP. The PKG generates all secret keys to every node in its trust domain. This kind of trust model leads to a single point of failure. The model will only work for networks where users trust the PKG because of the key escrow problem. This means that 1) the users do not care that the PKG can monitor traffic or 2) they trust that the PKG will not monitor traffic. In WSN this is not a problem. Typically networks that users might reacts to this kind of security structure could be telecommunication and email services. However, there is limited security in these types of network anyway. Telcos have full control over all data flowing through their servers and email actors such as Google states that their system is analyzing all content related to a Google user, including email [Gib14]. The problem is that these actors do not want to make all content opaque for themselves, because they use it for their business. My point being, if this is going to be the case anyway in the future, when the network switches over to NDN they could secure their systems with cryptography such as IBC to make it more difficult for adversaries to eavesdrop or perform any other form of attacks.

Another problem is key revocation. In ordinary PKI a device can create its own new key pair when compromised. This is not possible with IBC, because the PKG is needed to extract the SK. One suggestion has been to add a timestamp to the **Name** (e.g. monthly), but this introduces overhead for the PKG which has to renew

private keys for everybody each month. Also, one typically wants to reduce the time of vulnerability. Worst case, an adversary could act as an legit user for a month. With the FSM as a key distribution scheme, every user will be notified when an identity is revoked and replaced.

However, periodically renewal of keys might not be an issue in the HSS due to its natural size of devices participating in such a network. One does not always know when a SK_{device} is compromised, and thus periodic renewal of secret keys is a security measure that might be worth the cost.

Another problem occurs if the PKG must renew its key pair, MSK and MPK. Then secret keys for all devices have to be renewed.

A comparison of IBC and other solutions is shown in Table 7.1. In IBC, the number of keys a node has to manage is smaller than any other solution. There is no need for digital certificates, and it is not possible to forge a signature. Compromise of a single CEK, will only lead to compromise of the content which was encrypted using this key. Compromise of a SK, will only lead to compromise of the device which holds the corresponding ID.

	Key distribution	Number of keys	Key Directory	Digital certificate	Forward encryption	Nonrepudiation
Symmetric key cryptography	Problematic	$O(n^2)$	At each node	No	No	No
Random key predistribution	Simple	$O(n)$	At each node	No	No	No
PKC	Complex	$O(n)$	At each node and key center	Yes	No	Yes
IBC	Simple	N	No	No	Yes	Yes

Table 7.1: Comparison with PKC and IBC [PS12, Table 9.6].

7.3 Scalability

Distributing the ID-list can be an issue, as the list can grow linearly with the number of participants in the trust domain. However, this might not be a huge problem in the use case which is addressed in this thesis. Lets set an upper limit for the ID to be 20 bytes. Considering that the number of devices in the HSS will not grow larger than e.g. 100 devices, the list will almost be 2 kilobyte of data $((20B/1024) * 100 = 1.95KB)$. In a sensor network where the number of sensor can exceed 1 million sensors, the list of every ID can be cumbersome for each device. 1 million sensors will approximately

require 19 megabyte of data ($(20B/1024/1024) * 1000000 = 19.07MB$), which will have to be synchronized. However, in such sensor networks, each node does not need to know all other names, but rather the name its gateway, i.e. a sink node. Such logic scheme is not implemented or considered a part of this thesis.

7.4 Preshared Secret

The sharing of a temporary random key \mathbf{tk} to register a device is assumed to be preloaded in an offline mode or done in a controlled wired environment, i.e. not tapped. As mentioned, I suggest using NFC to perform the sharing of \mathbf{tk} . The NFC signal is hard to eavesdrop outside a radius of 1 meter, thus the sharing is assumed to be secure. However, in scenarios where a node cannot share a \mathbf{tk} in a physical manner, one have to rely on performing the device registration with some sort of asymmetric encryption. But this solution introduces the question “who is able to play the role as a device?”. The downfall is that it removes the authentication process that is performed when preloading the \mathbf{tk} .

7.5 Key Distribution

The FSM makes it possible for users to know who has a valid ID within the PKGs trust domain. One drawback with the key distribution scheme I have proposed, is DoS on **Sync Interest** and **Sync Data**. For the sender to be 100% sure that the message is encrypted with the latest ID, the sender has to rely on that it has received the latest sync state available from the PKG. Likewise when a receiver verifies a signature from another device in the trust domain, it has to rely on the same principle to be able to know if the belonging ID is still valid. In the case where an adversary has found a SK, the SK is compromised and the adversary can try to deny the distribution of a updated list, i.e. running a DoS on the **Sync Data**, from the distributor. This however, is a complicated attack. An updated list would spread fast in a large network. Performing DoS on every node is not easy, and would block the network access for the adversary anyway. In [SA99] Frank Stajano and Ross Anderson mentions possible DoS attacks, such as radio jamming and battery exhaustion. All applications that relies on some sort of crucial information derived using FSM (section 4.1) are vulnerable to this kind of DoS.

One question left to be solved is how expensive the **Sync Interest** would be for each device. The **Sync Interest** should be expressed periodically for always having the updated ID-list. In equivalent networks deployed over IP, **Hello** messages is often used, hence the **Sync Interest** can be modified to be a combination of **Hello** and **Sync Interest**.

7.6 Testing

The application is not tested with real sensors, hence I cannot conclude with anything regarding the computational power of such devices, nor the life time of the battery when performing IBE.

An interesting question is the performance difference of IBC versus RSA. In Table 6.3 we can see that IBC is performing better on key generation, signing and verification than regular asymmetric cryptography, RSA. However encryption and decryption of CEK is not performing better with the IBE schemes I use, but the difference is minimal compared to the results Xinwen Zhang et al. got in their implementation [ZCX⁺11], which was 1.7 seconds difference with IBE compared to RSA.

Looking at Table 6.4 and studying the Table 6.3, we can see that the time spent on each protocol can be correlated to the time spent doing IBE **Encrypting** CEK and IBE **Decrypting** CEK operations. We can see that the data pull on C1 takes about 61 ms. This protocol involves one IBE **Encrypting** CEK (41 ms) and one IBE **Decrypting** CEK (20 ms). Hence it seems that the IBC requires most of the time of this protocol. The results listed are done in a virtual environment, hence the total latency should be somewhat more in a deployed network.

7.7 Other Use Cases

The trust model used in the HSS can be used in any network where the issues of having a TTP is accepted. Such systems can for instance be:

1. Home automation systems
2. Building Automation System (BAS)
3. Building Management System (BMS)
4. Health care systems
5. Military networks
6. Sensor networks such as disaster, habitat and hazard monitoring

And thus this trust model using IBC over NDN could be widely used in the future.

Chapter 8

Conclusion and Future Work

In this chapter the conclusion of this thesis will be presented and the future work will be listed.

8.1 Conclusion

A new network protocol is much needed due to the lack of security in existing networks and the continually increase of data traffic around the world. This thesis explains the architecture of the proposed future Internet protocol NDN. NDN facilitates a lot of concepts that shows to be a huge benefit for todays Internet, and the predicted increase of IoT. The naming of content and content routing provides usability to IoT and WSN. The concept of IBC shows to be highly applicable to IoT and WSN, and running IBC over NDN, makes it even more practical. This is because of the content naming concept that NDN is built upon. It is easier to secure data, relate data to publisher, and authenticate that the publisher is aware of what content it published, than in an IP network. Using IBC in a WSN running over NDN makes applications with security less complex and more practical than using equivalent security such as RSA.

In this thesis I have developed an application thought to be deployed in a WSN, written in Python with IBC used for signing and verification, encryption and decryption. The system is running over the new network protocol called Named Data Networking. The work shows how applicable NDN together with IBC are for IoT. The application is tested to see how the suggested protocols for device registration and data pull performs with IBC. I have proven that my suggested proposal is a secure system that can easily be implemented, achieving confidentiality, integrity and authenticity, as well as trust.

8.2 Future Work

The implementation of the HSS does not include integration of the FSM, which is a part of the future work. The system is not tested on relevant sensors and devices to measure latency and performance.

An implementation of a full worthy IBC solution in PyNDN2 is not implemented. This implementation should include making IBC as a part of the PyNDN2 framework, so that developers easily can make use of IBE and IBS performing encryption, decryption, signing and verification.

The IBC schemes used in the Charm framework does not provide a scheme that implements IBE and IBS together in one scheme. This should not be a huge task to implement, but it will decrease the device registration round-trip time as well as minimizing the use of several keys, i.e. easier key management. It is explained how to derive a signature scheme from any IBE scheme [Wat09, Section 4].

References

- [AAG⁺13] Akinyele, Joseph A., Garman, Christina, Miers, Ian, Pagano, Matthew W., Rushanan, Michael, Green, Matthew, Rubin, and Aviel D. Charm: a framework for rapidly prototyping cryptosystems. 3(2):111–128, 2013.
- [ACM14] Marica Amadeo, Claudia Campolo, and Antonella Molinaro. Multi-source data retrieval in iot via named data networking. In *1st International Conference on Information-Centric Networking, ICN'14, Paris, France, September 24-26, 2014*, pages 67–76, 2014.
- [ADI⁺12] Bengt Ahlgren, Christian Dannewitz, Claudio Imbrenda, Dirk Kutscher, and Börje Ohlman. A survey of information-centric networking. *IEEE Communications Magazine*, 50(7):26–36, 2012.
- [AMS09] G. Appenzeller, L. Martin, and M. Schertler. Identity-Based Encryption Architecture and Supporting Data Structures. RFC 5408, RFC Editor, January 2009.
- [AMY⁺] Alexander Afanasyev, Ilya Moiseenko, Yingdi Yu, Wentao Shang, Lixia Zhang, Junxiao Shi, Yi Huang, Jerald Paul Abraham, Beichuan Zhang, Steve DiBenedetto, Chengyu Fan, Christos Papadopoulos, Davide Pesavento, Giulio Grassi, Giovanni Pau, Hang Zhang, Tian Song, Haowei Yuan, Hila Ben Abraham, Patrick Crowley, Syed Obaid Amin, Vince Lehman, Minsheng Zhang, and Lan Wang. Named data networking forwarding daemon. <http://named-data.net/doc/NFD/current/overview.html>. [Online; accessed 08-May-2015].
- [ASLF14] Younes Abidy, Bilel Saadallah, Abdelkader Lahmadi, and Olivier Festor. Named data aggregation in wireless sensor networks. In *2014 IEEE Network Operations and Management Symposium, NOMS 2014, Krakow, Poland, May 5-9, 2014*, pages 1–8, 2014.
- [ASZ⁺15] Alexander Afanasyev, Junxiao Shi, Beichuan Zhang, Lixia Zhang, Ilya Moiseenko, Yingdi Yu, Wentao Shang, Yi Huang, Jerald Paul Abraham, Steve DiBenedetto, Chengyu Fan, Christos Papadopoulos, Davide Pesavento, Giulio Grassi, Giovanni Pau, Hang Zhang, Tian Song, Haowei Yuan, Hila Ben Abraham, Patrick Crowley, Syed Obaid Amin, Vince Lehman, and Lan Wang. Nfd developer’s guide, ndn-0021. Technical report, February 2015. Revision 3.

- [BB04] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 443–459, 2004.
- [BDSZ94] Vaduvur Bharghavan, Alan J. Demers, Scott Shenker, and Lixia Zhang. MACAW: A media access protocol for wireless lan’s. In *SIGCOMM*, pages 212–225, 1994.
- [Ben07] LYNN Ben. *On the implementation of pairing-based cryptosystems*. PhD thesis, PhD thesis, Stanford University, 2007. <https://crypto.stanford.edu/psc/thesis.pdf>, 2007.
- [BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pages 213–229, 2001.
- [BGK12] Alexandra Boldyreva, Vipul Goyal, and Virendra Kumar. Identity-based encryption with efficient revocation. *IACR Cryptology ePrint Archive*, 2012:52, 2012.
- [BGNT14] Jeff Burke, Paolo Gasti, Naveen Nathan, and Gene Tsudik. Secure sensing over named data networking. In *2014 IEEE 13th International Symposium on Network Computing and Applications, NCA 2014, Cambridge, MA, USA, 21-23 August, 2014*, pages 175–180, 2014.
- [Bid06] Hossein Bidgoli. *Handbook of Information Security, Volume 2, Information Warfare, Social, Legal, and International Issues and Security Foundations*, chapter IBE (Identity-Based Encryption). 2006.
- [BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993.*, pages 62–73, 1993.
- [BZBH97] R. Braden, L. Zhang, S. Berson, and S. Herzog. Compressing TCP/IP Headers. RFC 2205, RFC Editor, September 1997.
- [CCGT13] Alberto Compagno, Mauro Conti, Paolo Gasti, and Gene Tsudik. Poseidon: Mitigating interest flooding ddos attacks in named data networking. *CoRR*, abs/1303.4823, 2013.
- [CGH04] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.
- [Cre08] Cas J. F. Cremers. The scyther tool: Verification, falsification, and analysis of security protocols. In *Computer Aided Verification, 20th International Conference, CAV 2008, Princeton, NJ, USA, July 7-14, 2008, Proceedings*, pages 414–418, 2008.

- [DGTU12] Steve DiBenedetto, Paolo Gasti, Gene Tsudik, and Ersin Uzun. Andana: Anonymous named data networking application. In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*, 2012.
- [DMS04] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, pages 303–320, 2004.
- [FAC14] Wenliang Fu, Hila Ben Abraham, and Patrick Crowley. isync: a high performance and scalable data synchronihttps://github.com/named-dataazation protocol for named data networking. In *1st International Conference on Information-Centric Networking, ICN'14, Paris, France, September 24-26, 2014*, pages 181–182, 2014.
- [Fou] National Science Foundation. Future internet architecture project. [Online; accessed 5-May-2015].
- [Gib14] Samuel Gibbs. Gmail does scan all emails, new google terms clarify. 2014.
- [GPR12] Sergio Gusmeroli, Salvatore Piccione, and Domenico Rotondi. Iot access control issues: A capability based approach. In *Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2012, Palermo, Italy, July 4-6, 2012*, pages 787–792, 2012.
- [GTU14] Cesar Ghali, Gene Tsudik, and Ersin Uzun. Network-layer trust in named-data networking. *Computer Communication Review*, 44(5):12–19, 2014.
- [GTUZ13] Paolo Gasti, Gene Tsudik, Ersin Uzun, and Lixia Zhang. Dos and ddos in named data networking. In *22nd International Conference on Computer Communication and Networks, ICCCN 2013, Nassau, Bahamas, July 30 - Aug. 2, 2013*, pages 1–7, 2013.
- [HKS15] Dennis Hofheinz, Jessica Koch, and Christoph Striecks. Identity-based encryption with (almost) tight security in the multi-instance, multi-ciphertext setting. In *Public-Key Cryptography - PKC 2015 - 18th IACR International Conference on Practice and Theory in Public-Key Cryptography, Gaithersburg, MD, USA, March 30 - April 1, 2015, Proceedings*, pages 799–822, 2015.
- [Jac88] Van Jacobson. Resource reservation protocol (rsvp). In *SIGCOMM*, pages 314–329, 1988.
- [Jac90] V. Jacobsen. Compressing TCP/IP Headers. RFC 1144, RFC Editor, February 1990.
- [Jac09] Van Jacobsen. Introduction to content centric networking, part 3. bremen, fiss 2009. 2009.
- [LN08] An Liu and Peng Ning. Tinyecc: A configurable library for elliptic curve cryptography in wireless sensor networks. In *Proceedings of the 7th International Conference on Information Processing in Sensor Networks, IPSN 2008, St. Louis, Missouri, USA, April 22-24, 2008*, pages 245–256, 2008.

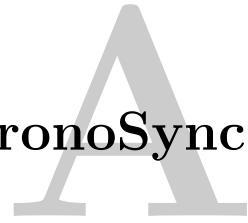
- [LV09] Benoît Libert and Damien Vergnaud. Adaptive-id secure revocable identity-based encryption. In *Topics in Cryptology - CT-RSA 2009, The Cryptographers' Track at the RSA Conference 2009, San Francisco, CA, USA, April 20-24, 2009. Proceedings*, pages 1–15, 2009.
- [LZZ⁺15] Qi Li, Xinwen Zhang, Qingji Zheng, Ravi Sandhu, and Xiaoming Fu. LIVE: lightweight integrity verification and content access control for named data networking. *IEEE Transactions on Information Forensics and Security*, 10(2):308–320, 2015.
- [Mø15] Haakon Garseg Mørk. Master thesis work. <https://github.com/haakonmo/master-thesis-ndn/tree/master/src>, 2015.
- [Nac05] David Naccache. Secure and *Practical* identity-based encryption. *IACR Cryptology ePrint Archive*, 2005:369, 2005.
- [NT] NDN-TEAM. Ndn packet format. <http://named-data.net/doc/ndn-tlv/>. [Online; accessed 02-May-2015].
- [NT15a] NDN-TEAM. Named data. <https://github.com/named-data>, 2015.
- [NT15b] NDN-TEAM. Pyndn2. <https://github.com/named-data/PyNDN2>, 2015.
- [OAM⁺07] Leonardo B. Oliveira, Diego F. Aranha, Eduardo Morais, Felipe Daguano, Julio López, and Ricardo Dahab. Tinytate: Identity-based encryption for sensor networks. *IACR Cryptology ePrint Archive*, 2007:20, 2007.
- [PS12] Harsh Kupwade Patil and Stephen A. Szygenda. *Security for Wireless Sensor Networks Using Identity-Based Cryptography*. Auerbach Publications, Boston, MA, USA, 1st edition, 2012.
- [Pyt] Python. Watchdog. <https://pypi.python.org/pypi/watchdog>. [Online; accessed 24-April-2015].
- [Rad] Jerome Radcliffe. Hacking medical devices for fun and insulin: Breaking the human scada system. Black Hat Conference 2011.
- [RL96] Ronald L Rivest and Butler Lampson. Sdsi-a simple distributed security infrastructure. *Crypto*, 1996.
- [Ros14] Gregory L. Rosston. Increasing wireless value: Technology, spectrum, and incentives. *JTHTL*, 12(1):89–112, 2014.
- [SA99] Frank Stajano and Ross J. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Security Protocols, 7th International Workshop, Cambridge, UK, April 19-21, 1999, Proceedings*, pages 172–194, 1999.
- [San14] Intelligent Broadbands Networks Sandvine. Global internet phenomena report 1h 2014, 2014.

- [SDM⁺14] Wentao Shang, Qiuhan Ding, Alessandro Marianantoni, Jeff Burke, and Lixia Zhang. Securing building management systems using named data networking. *IEEE Network*, 28(3):50–56, 2014.
- [SE13] Jae Hong Seo and Keita Emura. Efficient delegation of key generation and revocation functionalities in identity-based encryption. *IACR Cryptology ePrint Archive*, 2013:18, 2013.
- [Sha84] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, pages 47–53, 1984.
- [SJ09] Diana Smetters and Van Jacobson. Securing network content. Technical report, 2009. PARC Tech Report.
- [SK05] Aaditeshwar Seth and Srinivasan Keshav. Practical security for disconnected nodes. In *Proceedings of the First International Conference on Secure Network Protocols, NPSEC'05*, pages 31–36, Washington, DC, USA, 2005. IEEE Computer Society.
- [SNO13] Won So, Ashok Narayanan, and David Oran. Named data networking on a router: Fast and dos-resistant forwarding with hash tables. In *Symposium on Architecture for Networking and Communications Systems, ANCS '13, San Jose, CA, USA, October 21-22, 2013*, pages 215–225, 2013.
- [SOS⁺08] Piotr Szczechowiak, Leonardo B. Oliveira, Michael Scott, Martin Collier, and Ricardo Dahab. Nanoecc: Testing the limits of elliptic curve cryptography in sensor networks. In *Wireless Sensor Networks, 5th European Conference, EWSN 2008, Bologna, Italy, January 30-February 1, 2008, Proceedings*, pages 305–320, 2008.
- [Wat04] Brent R. Waters. Efficient identity-based encryption without random oracles. *IACR Cryptology ePrint Archive*, 2004:180, 2004.
- [Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, pages 619–636, 2009.
- [WCZ⁺14] Kai Wang, Jia Chen, Huachun Zhou, Yajuan Qin, and Hongke Zhang. Modeling denial-of-service against pending interest table in named data networking. *Int. J. Communication Systems*, 27(12):4355–4368, 2014.
- [YHB12] Yusnani Mohd Yussoff, Habibah Hashim, and Mohd Dani Baba. Identity-based trusted authentication in wireless sensor network. *CoRR*, abs/1207.6185, 2012.
- [ZA13] Zhenkai Zhu and Alexander Afanasyev. Let's chronosync: Decentralized dataset state synchronization in named data networking. In *2013 21st IEEE International Conference on Network Protocols, ICNP 2013, Göttingen, Germany, October 7-10, 2013*, pages 1–10, 2013.

- [ZAB⁺14] Lixia Zhang, Alexander Afanasyev, Jeff Burke, Van Jacobson, kc claffy, Patrick Crowley, Christos Papadopoulos, Lan Wang, and Beichuan Zhang. Named data networking. *Computer Communication Review*, 44(3):66–73, 2014.
- [ZCX⁺11] Xinwen Zhang, Katharine Chang, Huijun Xiong, Yonggang Wen, Guangyu Shi, and Guoqiang Wang. Towards name-based trust and security for content-centric network. In *Proceedings of the 19th annual IEEE International Conference on Network Protocols, ICNP 2011, Vancouver, BC, Canada, October 17-20, 2011*, pages 1–6, 2011.
- [ZEB⁺10] Lixia Zhang, Deborah Estrin, Jeffrey Burke, Van Jacobsen, James D. Thornton, Diana K. Smetteres, Beichuan Zhang, Gene Tsudik, kc claffy, Dmitri Krioukov, Dan Massey, Christos Papadopoulos, Tarek Abdelzaher, Lan Wang, Patrick Crowley, and Edmund Yeh. Named data networking (ndn) project, ndn-0001. Technical report, October 2010.

Appendix

ChronoSync



Since NDN provides multicast in the network layer as explained in Figure 2.5, we do not have to think of network load in the same way as in IP. To achieve distributed synchronization of a **Dataset**, the NDN-team has developed ChronoSync, a decentralized synchronization framework over NDN. ChronoSync assumes that a group of nodes knows the **Name** of a synchronization group, e.g. `/ndn/broadcast/FileSync-0.1/<group_room>/`. The synchronization application is built upon state digests, which is that each participating node stores a hash of its current **Dataset**. Each node in a ChronoSync application broadcasts its sync state in a **Sync Interest** (e.g. `/ndn/broadcast/FileSync-0.1/<group_room>/<state>`). When a node receives a **Sync Interest**, it will inspect the state of the **Interest**, and compare with its own state. Each node holds a state tree that is used to detect new and outdated states. If the incoming **Interest** state is equal to the receiving node's state, the node has no reason to do anything, as the system is in a *stable state* from the node's point of view. If not, the receiving node has to find out whether the incoming **Interest** is 1) a state the node itself has been in, or if its 2) a new state. In case of 1), the receiving node has new **Data** and should provide the new content as a response to the incoming **Interest**. In case of 2), the receiving node should send out a **Recovery Interest** for the new state.

1. *Sync Interest* is an **Interest** that a participating node sends out to discover new **Data**.
2. *Sync Data* is a response to 1), if a participating node has new **Data**.
3. *Recovery Interest* is an **Interest** sent out if a node discovers that another node has a newer state.
4. *Recovery Data* is a response to 3).

When the group is in a stable state, each Sync **Interest** is equivalent, hence only one entry at each router's PIT is created, forming a temporary multicast tree. This **Interest** is periodically sent out from each subscriber maintaining the multicast tree, resulting in that the producer has the possibility to answer the Sync **Interest** with Sync **Data** whenever the producer has a new **Dataset**.

ChronoSync is only taking care of **Data** discovery, and leaves other logic to the application that is using ChronoSync. Such logic can be e.g. what should happen when a new participant enters the room. Should all history be downloaded? Or who is allowed to publish content in each synchronization group?

ChronoSync is explained in detail here [ZA13].

Appendix B

Formal Security Analysis Code

B.1 Scyther Security Analysis of Device Registration

Scyther is a tool for verifying the security in protocols. The security verification for the device registration phase 2 is modeled in the SPDL code presented in Listing B.1. To better understand the SPDL code, Table B.1 presents the mapping of Figure 5.3 to the code.

Figure	SPDL Code	Description
ID_d	D	Identity of the device
ID_{PKG}	PKG	Identity of the PKG
n	R	Random nonce
sk	SK	Secret key to an identity
$c_1 = \text{AES_Enc}_{tk}[ID_d n]$	$c_1 = \{ D, R \}_k(\text{PKG}, D)$	AES encrypted content
$c_2 = \text{AES_Enc}_{tk}[sk \hat{n}]$	$c_2 = \{ SK, R \}_k(\text{PKG}, D)$	AES encrypted content
$s = \text{Sign}(\text{mpk} sk_{\text{pkg}} c_2)$	$s = \{ \text{SHA1}(c_2) \}_{sk}(\text{PKG})$	Signature

Table B.1: Mapping of the Figure 5.3 and the SPDL code.

```

1 hashfunction SHA1;
2 usertype SecretKey;
3
4 const pk: Function;
5 secret sk: Function;
6 inversekeys (pk, sk);
7
8 macro c1 = { D, R }k(PKG, D);
9 macro c2 = { SK, R }k(PKG, D);
10 macro s = { SHA1(c2) }sk(PKG);
11 protocol HSSDeviceRegistration (PKG, D)
12 {
13
```

70 B. FORMAL SECURITY ANALYSIS CODE

```

14  role PKG
15  {
16    var R: Nonce;
17
18    recv_1(D, PKG, c1);
19
20    fresh SK: SecretKey;
21
22    send_2(PKG, D, c2 , s );
23
24    claim_PKG1(PKG, Alive);
25    claim_PKG2(PKG, Secret, SK);
26    claim_PKG3(PKG, Weakagree);
27    claim_PKG4(PKG, Niagree);
28    claim_PKG5(PKG, Nisynch);
29  }
30
31  role D
32  {
33    fresh R: Nonce;
34
35    send_1(D, PKG, c1);
36
37    var SK: SecretKey;
38
39    recv_2(PKG, D, c2 , s );
40
41    claim_D1(D, Alive);
42    claim_D2(D, Secret, SK);
43    claim_D3(D, Weakagree);
44    claim_D4(D, Niagree);
45    claim_D5(D, Nisynch);
46  }
47 }

```

Listing B.1: Device Registration SPDL

B.2 Scyther Security Analysis of Data Pull

Scyther is a tool for verifying the security in protocols. The security verification for the data pull is modeled in the SPDL code presented in Listing B.2. To better understand the SPDL code, Table B.2 presents the mapping of Figure 5.4 to the code.

Figure	SPDL Code	Description
ID_d	D	Identity of the device
ID_m	M	Identity of the mobile
n	R	Random nonce
sk	SK	Secret key to an identity
$c_cek = \text{Encrypt}(\text{mpk} \parallel ID_d \parallel \text{cek})$	$ccek = \{ \text{cek} \}pk(D)$	IBEncrypted CEK
$c = \text{AES_Enc}_{cek}(\text{data} \parallel \tilde{n})$	$c = \{ \text{data}, R \}k(M,D)$	AES encrypted content
$m_1 = (ID_d \parallel n \parallel \text{request})$	$m1 = (M, D, R)$	Message
$c_2 = (c_cek \parallel c)$	$c2 = (M, c, ccek)$	Message
$s_1 = \text{Sign}(\text{mpk} \parallel sk_m \parallel c_1)$	$s1 = \text{SHA1}(m1) sk(M)$	Signature
$s_2 = \text{Sign}(\text{mpk} \parallel sk_d \parallel c_2)$	$s2 = \text{SHA1}(c2) sk(D)$	Signature

Table B.2: Mapping of the Figure 5.4 and the SPDL code.

```

1 hashfunction SHA1;
2 usertype ContentEncryptionKey, Content;
3
4 const pk: Function;
5 secret sk: Function;
6 inversekeys (pk,sk);
7
8 macro m1 = (M, D, R);
9 macro s1 = { SHA1(m1) }sk(M);
10 macro ccek = { cek }pk(D);
11 macro c = { data, R }k(M,D);
12 macro c2 = ( M, c, ccek);
13 macro s2 = { SHA1(c2) }sk(D);
14
15 protocol HSSDataPull (M, D)
16 {
17   role M
18   {
19     fresh R: Nonce;
20
21     send_1(M, D, m1, s1);
22
23     var cek: ContentEncryptionKey;

```

72 B. FORMAL SECURITY ANALYSIS CODE

```

24     var data: Content;
25     recv_2(D, M, c2, s2 );
26
27     claim_M1(M, Alive);
28     claim_M2(M, Secret, cek);
29     claim_M3(M, Secret, data);
30     claim_M4(M, Weakagree);
31     claim_M5(M, Niagree);
32     claim_M6(M, Nisynch);
33 }
34
35 role D
36 {
37     var R: Nonce;
38
39     recv_1(M, D, m1 , s1 );
40
41     fresh cek: ContentEncryptionKey;
42     fresh data: Content;
43     send_2(D, M, c2, s2 );
44
45     claim_D1(D, Alive);
46     claim_D2(D, Secret, cek);
47     claim_D3(D, Secret, data);
48     claim_D4(D, Weakagree);
49     claim_D5(D, Niagree);
50     claim_D6(D, Nisynch);
51 }
52 }

```

Listing B.2: Data Pull SPDL