NTNU – Trondheim
Norwegian University of
Science and Technology

Identity-Based Trust Model in Named Data Networking

Haakon Garseg Mørk

Submission date: May 2015
Responsible professor: Stig F. Mjølsnes, ITEM
Co-supervisor: N/A

Norwegian University of Science and Technology
Department of Telematics

Abstract

Abstract goes here...

Acknowledgments

Contents

List of Figures

Listings

List of Tables

List of Acronyms

ACL Access Control List.

AES Advanced Encryption Standard.

ARP Address Resolution Protocol.

ASP Application Service Provider.

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.

DOI Digital Object Identifier.

DoS Denial of Service.

FIA Future Internet Architecture.

FIB Forwarding Information Base.

FSM File Synchronization Module.

HLR Home Location Register.

HSS Health Sensor System.

HTTPS Hypertext Transport Protocol Secure.

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.

IRTF Internet Research Task Force.

KB Kilobyte.

LAN Local Area Network.

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.

OS Operating System.

PARC Palo Alto Research Center.

PBC Pairing-Based Cryptosystems.

PIT Pending Interest Table.

PKG Private Key Generator.

PKI Public Key Infrastructure.

PyNDN2 Named Data Networking Client Library in Python.

RIB Routing Information Base.

SDSI Simple Distributed Security Infrastructure.

SHA1 Secure Hash Algorithm 1.

SSN Social Security Number.

TCP Transmission Control Protocol.

TLS Transport Layer Security.

TMPK Temporary Master Public Key.

TTP Trusted Third Party.

UCLA University of California, Los Angeles.

UDP User Datagram Protocol.

URI Uniform Resource Identifier.

URL Uniform Resource Locator.

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 a tremendous amount of work done 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 Rivest and Lampson Simple Distributed Security Infrastructure (SDSI) proposal [RL96], and subsequent work, that may be revisited for applicable to NDN.

1.2 Problem and Scope

When designing a new network protocol for the future Internet, one of the most significant changes should be security. NDN is being designed with security in mind. 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 the solution is probably not the one or the other, but rather case specific.

I address the trust management issue in a though sensor device network, e.g. a health sensor network. By using the Named Data Networking Forwarding Daemon (NFD) I will implement my proposal for such a sensor network over NDN.

1.3 Methodology

First I design the application flow in sequence diagrams. Based on how NDN is designed, I try to implement the proposed design and see where to redesign the application proposal for achieving minimal communication overhead, maximum security (i.e. Confidentiality, Integrity and Availability (CIA)) and usability.

1.4 Outline

This paper will first introduce one of the proposed protocols for the future Internet, NDN. I will explain the architecture of NDN as well as some related work regarding my application proposal. The application modules will be explain in detail and implementation choices will be discussed. At last I will present the results of the implementation and my conclusion of the trust model the application uses.

1.5 Notations

Notations used throughout in this thesis is listed in Table 1.1.

Symbol	Description
MPK_i	Master Public Key belonging to i
MSK_i	Master Secret Key belonging to i
SK_i	Secret Key belonging to i
PK_i	Public Key belonging to i
ID_i	Identity belonging to i

Table 1.1: Notations used throughout the thesis.

Chapter 2
Background

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

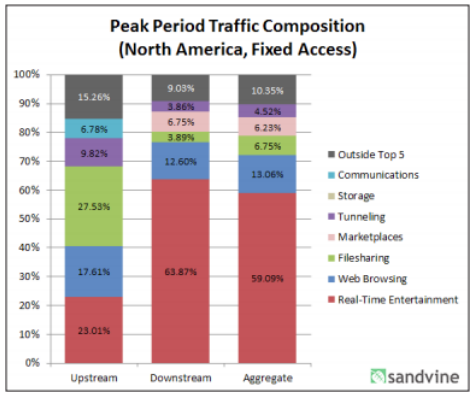
— Van Jacobsen, *SIGCOMM 2001*

This chapter will give a brief overlook of the motivation for Information-Centric Networking (ICN), as well as explaining more details of ICN protocols such as Content Centric Networking (CCN) and NDN. The NDN architecture will be reviewed. Finally a quick summary of related works.

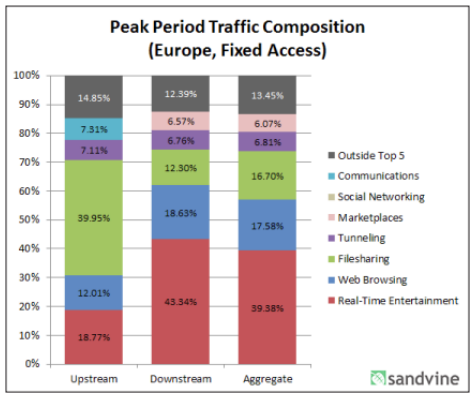
2.1 Motivation for Information-Centric Networking

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 architecture. As Internet have developed, the traffic has increased enormously over the past few years. In the Global Internet Phenomena Report 1H2014 done by Sandvine [San14], 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. With this in mind, the IP architecture does not provide an efficient transport model for what we are actually using the network for.

Designing the IP network, the security was not the first priority. A logical though considering that they did not know what the Internet was going to be used for and how big it has become. Most of the protocols related to Internet have been designed and deployed mainly with the goal of functionality to work. In the years after the birth of Internet, they soon found out that they needed security at several layers, due to that the application requirements and the transmission importance increased. Internet Protocol Security (IPsec) is a very good example of work trying to patch up security flaws in Internet.



(a) North America



(b) Europe

Figure 2.1: (a) Peak Period Aggregate Traffic Composition - North America, Fixed Access [San14]. (b) Peak Period Aggregate Traffic Composition - Europe, Fixed Access [San14].

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.The Internet of Things (IoT) trend is coming, and the IP network is not designed for broadcast and therefore wireless connection is not as easy as it should be. Devices should easily be able to communicate directly with each other without having to interconnect through a router.

find a cite

Another problem is the network redundancy. Looking at Figure 2.1 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 due to that 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. This makes every node dumb, hence the network is designed for redundancy when it comes to content download.

These design failures are some the reasons why the research for 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. 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, 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 and TCP header compression.

cite

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 have contributed to. One of the biggest contributers is UCLA, with Lixia Zhang in the lead. The NDN project is also one of few projects selected by National Science Foundation (NSF) 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 paper 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

a section explaining NDN briefly so the reader has something to relate to reading the following sections

2.3.2 Based on Existing Solution

The goal for the network design is essentially making it more 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 we have 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 from 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.
- Content Distribution Network (CDN) - Many Application Service Provider (ASP)s, such as Netflix and YouTube, have found out that the IP network performs a lot better for their costumers if they cache up their data.

write something about this

Namespace-based trust introduced in SDSI, binding names to public keys.

2.3.3 Packets

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

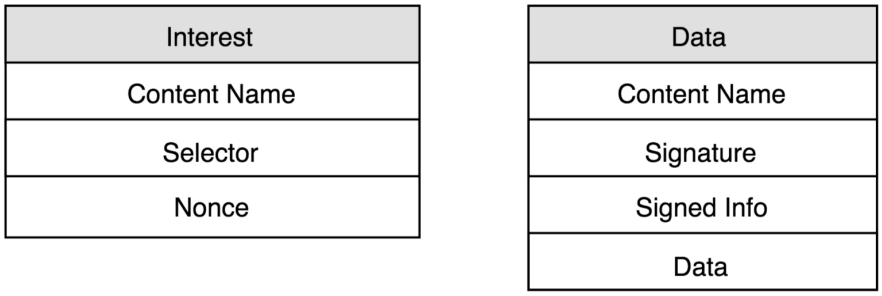


Figure 2.2: Interest packet and Data packet

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 does not contain "/ndn/no/ntnu/item".

- MustBeFresh - if True, a node cannot answer with a Data packet where the FreshnessPeriod has expired.

- 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 it 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. Now when somebody 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 only a Data packet can exists 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 the NDN network there is no strict rules for a Name. 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 greatest version.

When designing applications for the NDN network, one might learn from DNS and Operating System (OS) naming. write more

2.3.5 Network Node

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. 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 a 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.

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.

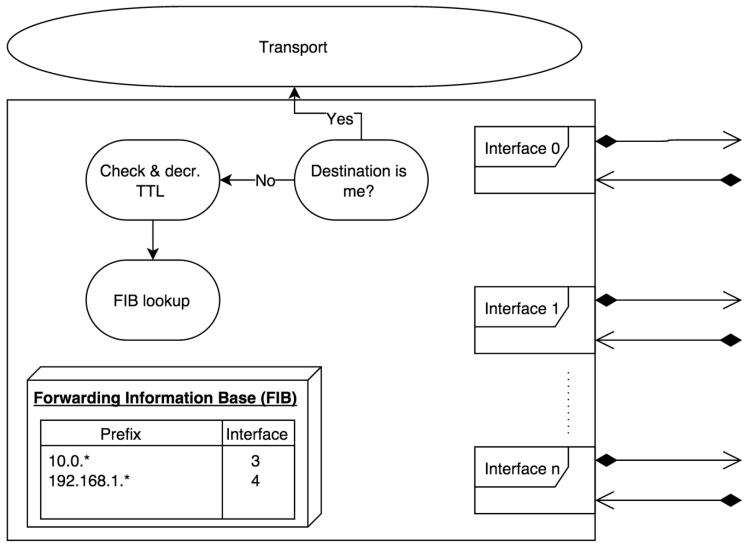


Figure 2.3: Model of IP node. A packets enters the node through an Face. The node decides whether the packet is for the node itself, or passes it further to next node, found in the FIB.

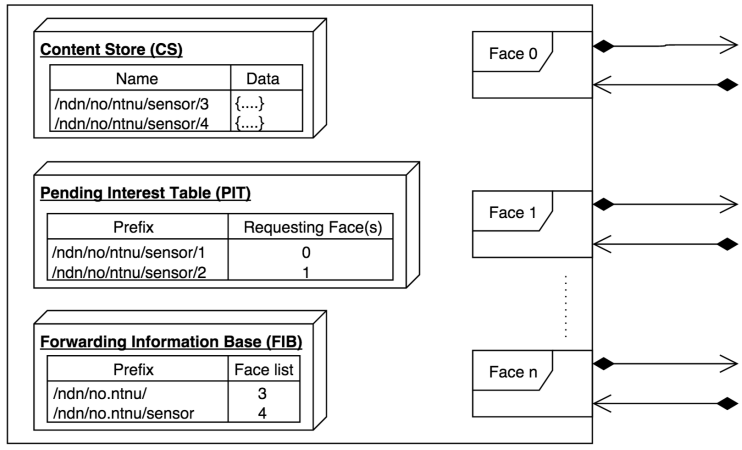


Figure 2.4: Model of NDN node. A packet enters through an Face. The node checks whether the Interest is already queried in the PIT, or stored in the CS, or passes it further to next node, found in the FIB.

Unicast (TcpFace, UdpFace) vs Multicast (MulticastUdp, Ethernet)

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. PIT), and 2) *what* has been satisfied earlier and still available, i.e. still cached inCS. With this ability the network can now supply requests with content already stored in cache, hence the network can naturally offer multicast. Figure 2.5 illustrates a NDN network where we can see that the network does not nearly have to send equal amount of traffic 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. When the second computer expresses the same Interest (3), the consecutive node already has cached the Data response matching to the Interest in its CS, hence the Interest is satisfied already at this point, and not forwarded any further. Same happens when the third computer expresses again the same Interest (4) to the network. Given that the file these computers are interested in is 4 gigabyte, the network saves a lot of traffic with multicast.

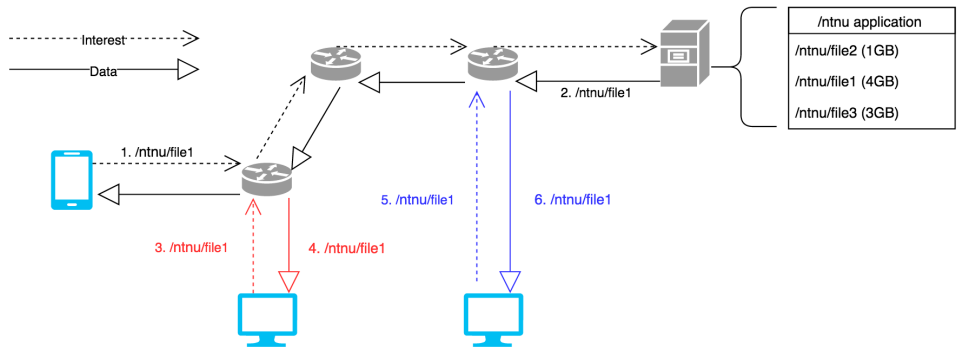


Figure 2.5: Multicast in NDN.

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. How to forward a Interest; routing strategy. A strategy per PIT entry. I.e. whether, when, and where to forward the Interest.

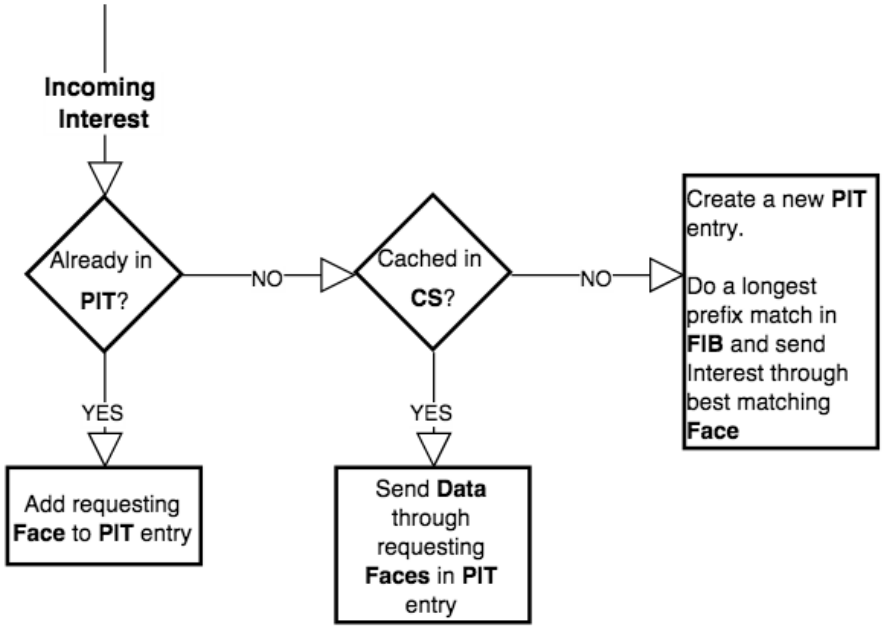


Figure 2.6: Decision tree for a NDN node when receiving an Interest.

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. 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).

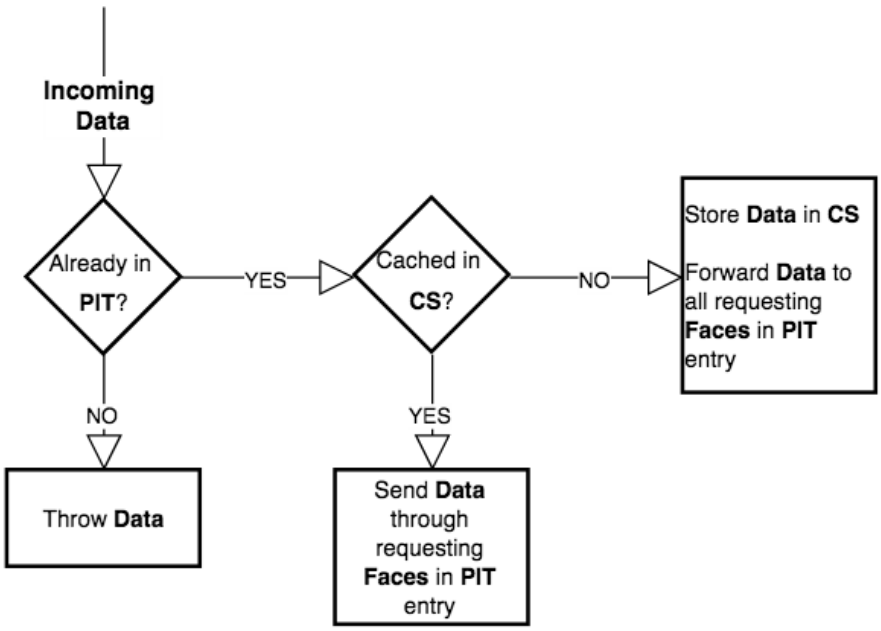


Figure 2.7: Decision tree for a NDN node when receiving Data.

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.

this section needs to be refactored

Trusting Host vs Trusting Content

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

The content is rarely encrypted and the confidentiality is not preserved, unless there is established a secure channel using e.g. Transport Layer Security (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. The solution is to encrypt the connection. This is the concept of securing the channel, and the trust is based on certificates. 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 over IP to retrieve content from other than the trusted host because you do not trust any other than the host you are connected to via the 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. 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 eyes (if necessary). This introduces a possibility that anyone can retrieve any piece of data from anyone regardless of secure channel or not. 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?

One can do hash verification on the content to be sure that the content is unmodified. But there should be a binding between the Name to the content. However this does not provide provenance nor relevance. For this there should be a linkage between the publisher, the Name and the content. Doing 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, and it provides 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. Now anybody can retrieve $\mathtt{M_{(N,P,C)}}$, hence an untrusted host and an insecure channel is not so bad anymore.

$$M_{(N,P,C)} = (N, C, Sign_P(N, C)) \tag{2.1}$$

A clear benefit of this approach is that the Name can be of any form. 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 whole network is able to decide the whole packet path, hence 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 data 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 capability of the network.

more in this section

2.4 Attacks

Paolo Gasti et al. identifies several DoS attacks on NDN in their paper about DoS & 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. proposes 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].

2.5 Related work

The work in this thesis builds upon three main concepts; synchronization, sensor networking and Identity-Based Cryptography (IBC). Some related work done will shortly be presented in this section.

2.5.1 Synchronization

Synchronization application over NDN called iSync [FAC14]. A synchronization application build by the NDN team is ChronoSync [ZA13].

2.5.2 Secure Data Retrieval from Sensors

Amadeo et al. [ACM14] proposes 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 closest sensor, e.g. the light in *this* room. In [ASLF14] Abid et al. simulate data aggregation in wireless sensor networks. Burke et al. addresses efficient and secure Sensing over NDN in [BGNT14]

Securing Building Management Systems Using Named Data Networking [SDM+14]

2.5.3 Identity-Based Cryptography in Named Data Networking

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

write more, mer utfyllende

Chapter 3
File Synchronization over Named Data Network

This chapter will present ChronoSync, and explaining what the File Synchronization Module (FSM) is built upon and its purpose over the NDN network.

3.1 ChronoSync

Since NDN provides multicast in 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>/`. 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).

All traffic in NDN is pull based, meaning that a user has to request the data for it to be sent. 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 three. This Interest is periodically sent out from each subscriber maintaining the multicast three, resulting 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 for instance; what should happen when a new participant enters the room. Should all history be download?

ChronoSync is explained in detail here [ZA13].

3.2 File Synchronization Module

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 owner. Each node always wants to have 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. Lets 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 wants 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 new 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. 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 3.1 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.
2. `a`, `b` and `c` ask for subscription to this list, and somehow authenticates them selves to `d` if confidentiality is required.
3. `d` approves those who should be approved, and returns a symmetric synchronization key. Again only 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.

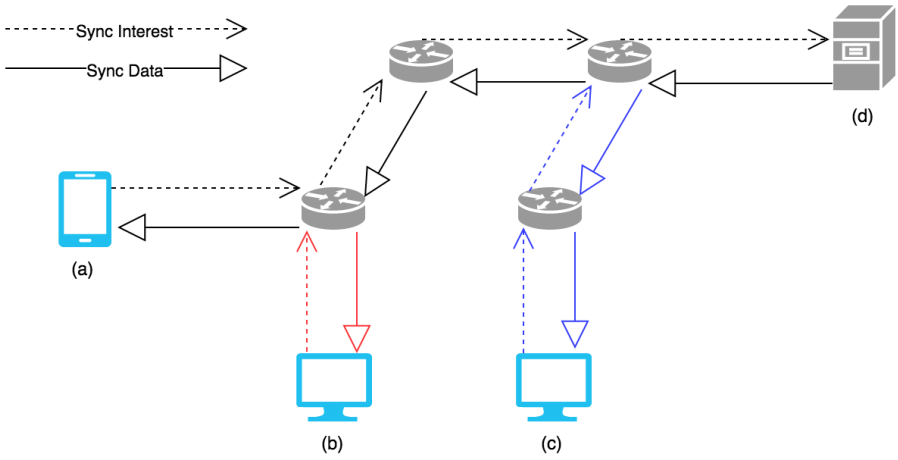


Figure 3.1: File Synchronization in NDN.

There are some issues that could occur. DoS on Sync Interest and Sync Data. If the FSM is used to revoke public keys as suggested, an attacker who has found the compromised private key, can try to deny the distribution of the new list, i.e. the Sync Data, from the distributor.

write more

Chapter 4
Key Infrastructure

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

4.1 Identity-Based Cryptography

IBE was first proposed by Shamir [Sha84] in 1984. The concept of IBE builds upon every user having an Identity (ID) that is used as the public key. This ID can be anything, i.e. email, phone number, Social Security Number (SSN), or a Name (subsection 2.3.4). This eliminates the need of certificates. Shamir did propose a scheme for IBS, but not a scheme for IBE. 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 with a random oracles model [Wat04], hence less practical.

IBE is based upon performing cryptography with a publicly know ID. Since the ID can be practically anything it is highly applicable for NDN where the ID can be a Name ("/ndn/no/ntnu/haakon"). Hence the Name becomes the public key.

There is a TTP in IBE that is called Private Key Generator (PKG). The PKGs task is to produce a secret key that corresponds to a given ID and provide

1. `Setup()` generates a key pair, Master Public Key (MPK) and Master Secret Key (MSK). These keys are used by only the PKG to extracting secret keys, encryption and decryption.
2. `Extract(`MPK_{PKG}`,` MSK_{PKG}`,` $\text{ID}_{\text{device}}$`)` generates a secret key from a given ID.
3. `Encrypt(`MPK_{PKG}`,` $\text{ID}_{\text{device}}$`, message)` encrypts the message.

4. `Decrypt(`MPK_{PKG}`,` SK_{device}`, cipher)` decrypts the cipher generated from the encryption.
5. `Signing(`MPK_{PKG}`,` SK_{device}`, message)` signs a hash digest of the message (e.g. Secure Hash Algorithm 1 (SHA1)).
6. `Verify(`MPK_{PKG}`,` ID_{device}`, message, signature)` verifies the signature.

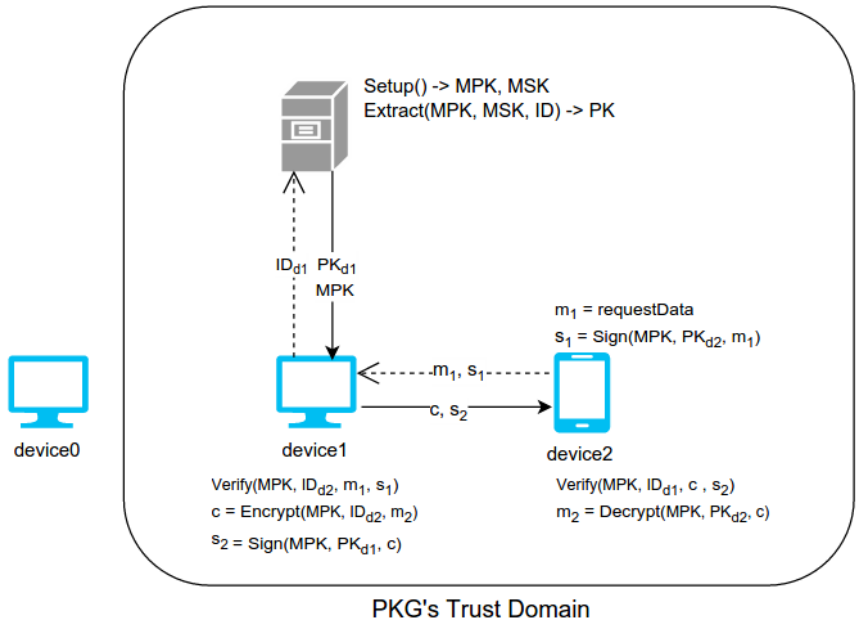


Figure 4.1: Methods of an IBC systems illustrated in action.

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.

Some drawbacks related to IBE are listed below:

1. If PKG is compromised. Adversary has secret key to all nodes that used the compromised PKG
2. PKG can read and write messages related to the node, because it has all secret keys, i.e. Man In The Middle (MITM).
3. PKG and the requesting node has to establish a secure channel.

4.2 Identity-Based Cryptography - Secureness

A random oracle is like a "black box" that outputs truly random numbers. When designing protocols in cryptography one first usually designs an ideal system where all parties have random oracle access, then proves the security. 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 just 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.

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 2014 Boneh and Boyen proposed a fully secure scheme in the standard model [BB04]. However it is not efficient.

First practical scheme was [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.

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

To understand the mathematical assumptions for IBE, the reader should take a look at [BF01, section 3] for details about bilinear maps and Bilinear Diffie-Hellman Problem (BDH).

4.3 Key Distribution

Instead of in PKI where each public key is signed by a certificate authority and the generated certificate is sent as a response in Hypertext Transport Protocol Secure (HTTPS) then validated by the the client, I want to make the certificate authority obsolete by distributing every ID (public key) issued by the PKG. This can be done through an IDSync application built upon the FSM presented in section 3.2. 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.

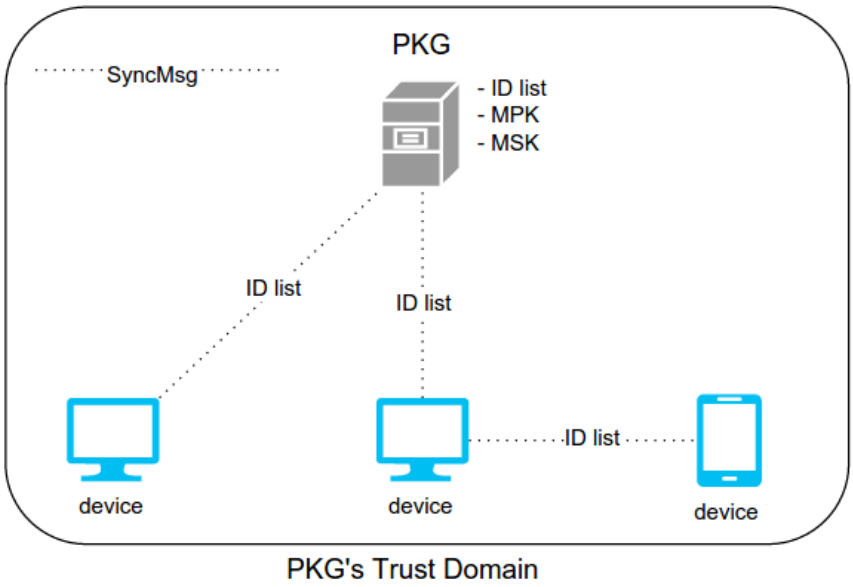


Figure 4.2: IDSync with tree devices and a PKG.

more existing solutions

4.4 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 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 the revocation solution is that it is cumbersome for the PKG. Boldyreva et al. proposes an revocation scheme [BGK12] based on efficient key-update, which makes the workload for the PKG a lot easier.

[SE13]

..something more

With the IDSync distribution

Chapter 5
Application

In this chapter the sensor application will be presented. Several sequence diagrams will be explained, as well as concepts needed to understand the flow of the application.

5.1 Health Sensors

There is an ongoing discussion of when the health technology revolution will come to human bodies now that IoT have 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 out of a substance, or if its 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 there 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 Health Sensor System (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 as everybody else that do not have diabetes, he does not want to manually monitor his glucose levels and adjust the insulin pump at 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.

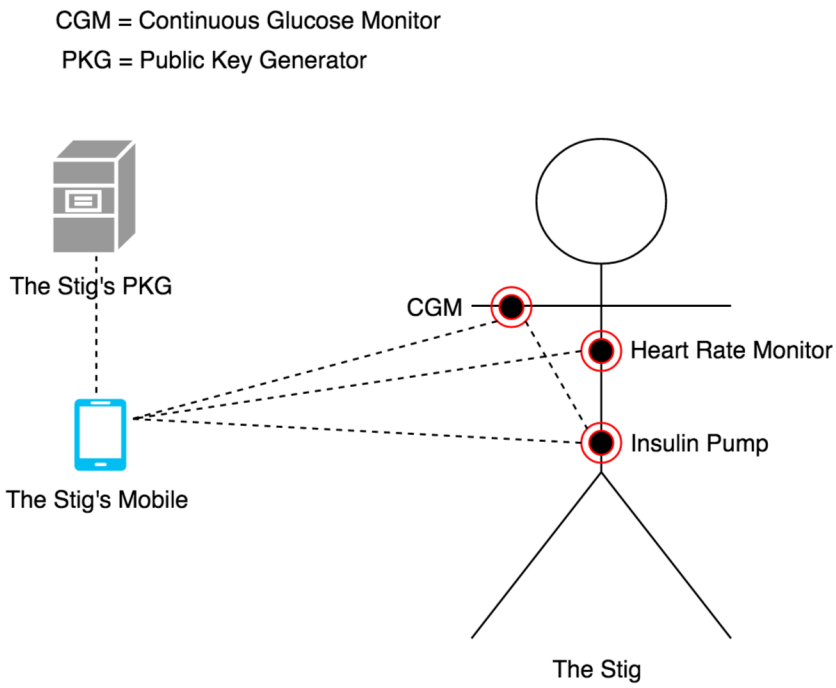


Figure 5.1: Health Sensor System

5.2 Health Sensor System

To have a secure system, it needs to be established trust 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. In IoT, we have in most cases the advantage of identifying devices in a physical matter. 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 matter and do not discuss whether how this should be done.

Also there is the concept of human-computer authentication [GRS05, JW05, Wei05]. This authentication method uses a shared secret before authentication.

5.2.2 Initialization

The goal for the initialization 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. The response message containing the secret key has to be 1) encrypted. This can be achieved by using asymmetric encryption on a CEK from which is used to do symmetric encryption on the secret key. The response message has to be 2) signed by the PKG for integrity and authenticity reasons. For it to make sense applying 3) a nonce for replay protection, the communication between the device and the PKG has to be unique of some kind. This implies that the device has to authenticate itself in a way that an adversary cannot do, e.g. a shared secret. This secret can for instance be a hardware implemented secret in the device that the user reads (from the package) and authenticates manually at the PKG before the initialization.

When The Stig is setting up his HSS, first he wants 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 PKG creates two 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)".

In Figure 5.2 the device plays the role as the mobile. The ID of every device is a part of the Name under which the device registers a prefix. I.e. a device register `/ndn/no/ntnu/<device>/<resource>` and hence its ID is `/ndn/no/ntnu/<device>`. To be able to communicate securely under initialization, the device have to create a temporary MPK and MSK and then extract a temporary secret key for its ID. At first, the device has to act as a PKG for itself to ensure confidentiality, and the trust between the device and the real PKG is based on the concept explained in subsection 5.2.1. The device sends an Interest appending the temporary MPK and the ID to the PKG asking to join the PKGs domain. The PKG encrypts a CEK with the temporary MPK and the ID received from the device. Then the PKG extracts the secret key for the device (this will be the key belonging to the PKGs trust domain) and uses the CEK to do a symmetric Advanced Encryption Standard (AES) encryption on the secret key. The Data packet response to the initialization Interest will contain the identity-based encrypted CEK, the symmetric encrypted secret key and the PKGs MPK. To finish the initialization protocol, the device decrypts the CEK, throws away the temporary keys, and finally decrypts the secret key. The device has established a trust with its PKG and can verify other

devices in this domain.

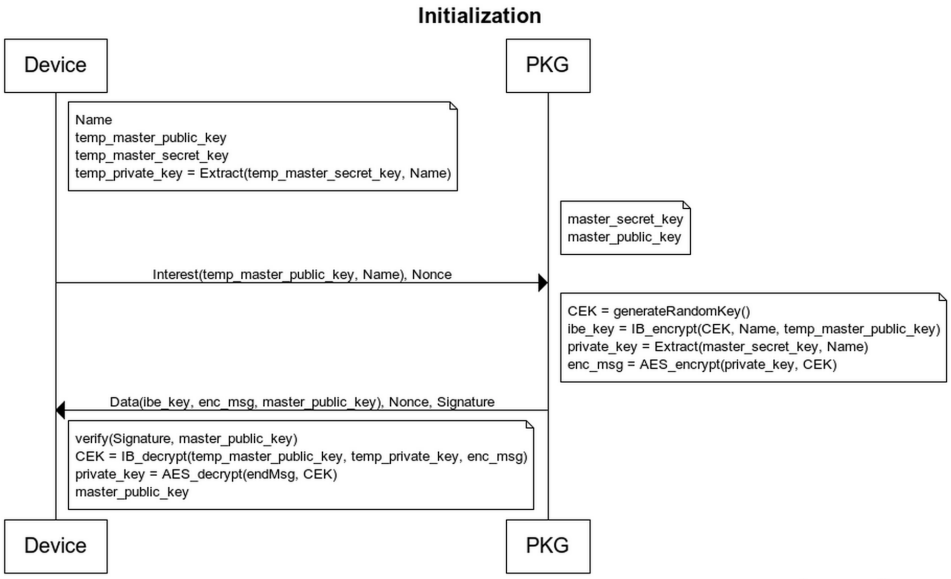


Figure 5.2: Initialization IBE

Now that the mobile is authenticated, devices can connect to the mobile through e.g. NFC for initialization. This results in a rendezvous authentication between the device and the mobile, and if the mobile is given the authorities to perform initialization (subsection 5.3.1), the new device can join the PKGs trust domain.

prove secureness of initialization protocol

5.2.3 Data Pull

The goal for this 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 initialized with the same PKG) and 2) the requester has to have granted access rights for the resource requested.

As illustrated in Figure 5.1, the device has joined the PKGs trust domain and are ready to communicate with other devices. This flow is illustrated in Figure 5.3. 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 receiver 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 receiver responds with the Data containing the resource. The receiver 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 and verifies the signature and decrypts the sensor data.

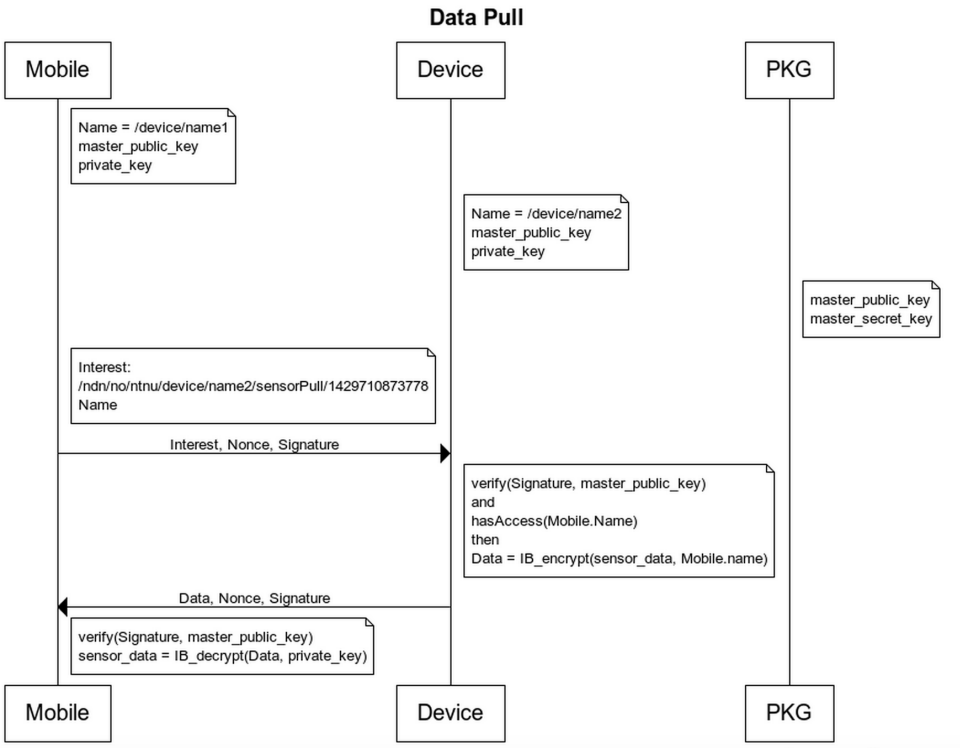


Figure 5.3: Mobile performing a data pull from a device in the network.

prove secureness of data pull protocol

5.2.4 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 public keys, i.e. every devices' ID. The distribution of this list can easily be achieved by using the FSM (section 3.2 & section 4.3). The PKG will be the distributor in this synchronization and each device will be a subscriber.

5.3 Trust Model

In this section the trust model will be explained.

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. since When a device retrieves an Interest for its sensor data, there should be an authorization mechanism. One solution for this is the Capability Based Approach to IoT Access Control [GPR12]. This design 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 have granted delegation rights to a mobile, and the mobile is not found trustworthy after a while, 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.

One can argue that once a device has been authenticated in the PKGs trust domain, everyone in the domain can be sure that the device will not abuse the information or functionalities available. However, due to scalability 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 [GPR12].

Another solution is to go for a Access Control List (ACL) based approach equivalent to what they did in [SDM+14].

5.4 Security Analysis

In this section the security of the protocols presented in subsection 5.2.2 and subsection 5.2.3 will be proven. Security analysis code in section A.1

!!

5.4.1 Confidentiality

The encryption can be achieved by encrypting with the Name of the requester. As explained in the sequence diagrams presented in the above sections, each Interest appends the requesters Name, hence all packets can be encrypted, and thus the confidentiality in the system is achieved.

5.4.2 Integrity and Authenticity

Each device will obtain a secret key allocated by its superior PKG, as explained in section 4.1. With the concept from subsection 5.2.1 together with the PKGs MPK, you can trust that the device is authorized for the PKGs trust domain. Hence all signed packets can be verified by anyone with the MPK. Every Interest has a timestamp attached to the Name (e.g. `/ndn/no/ntnu/device/name2/sensorPull/1429710873778`), i.e. milliseconds from `UTC 1970-01-01 00:00:00`, that is used for protection against replay attack.

5.4.3 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 of work and might run out of battery fast.

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 various application modules design and implementation choices will be explained. It will also be presented how these applications will be tested.

6.1 Installing Named Data Networking Protocol

There are several programs that is required to install for experimenting in a NDN environment. Installation guides can be found at the Github project [NT15a].

Named Data Networking C++ library with eXperimental eXtension (ndn-cxx) is a implementation of NDN primitives. It is a fundamental framework that NDN application requires.

NFD [AMY+] 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.

write more on this

NFD is running NDN over IP.

6.1.1 PyNDN2

The work done in this thesis is written in Python, hence the Named Data Networking Client Library in Python (PyNDN2) [NT15b] is used. This is an easy to use implementation of NDN and comes with example code.

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 signature when the Data packet is encoded and decoded.

6.2 Installing Identity-Based Cryptography

To be able to run IBC there is a Pairing-Based Cryptosystems (PBC) [Ben07] that needs to be installed. Further I use a Python implementation [AAG+13] of IBC.

The code in [Mø15, identityBasedCrypto.py] implements two IBE schemes and one IBS scheme.

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.

6.2.1 Charm - A Framework for Rapidly Prototyping Cryptosystems

The Charm framework [AAG+13] implements several IBE and IBS schemes in Python. Some small modifications had to be done in the Waters-IBS [Wat04] implementation in Charm. Not really something worth mentioning, yet explained due to the concept of academic reproduction. 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 (section 4.1), and not by another device 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`.

6.3 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 3.2. 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]

6.3.1 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] The code can be reviewed i [Mø15, fileSyncBuf.proto].

Init Interest - The MPK as well as the joining nodes Name is added in the KeyLocator. See Figure 6.1.

Sync Interest - Figure 6.2

Sync Data - Figure 6.2

Init Interest

Content Name	KeyLocator	MustBeFresh
/ndn/broadcast/FileSync-0.1/<sync_folder>	/ndn/no/ntnu/<source>/<ibeAlgorithm>/<mpk>	True

Figure 6.1: Initialization Interest for joining a synchronization folder.

6.3.2 File Watcher

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].

[1] Google Protocol Buffers - https://developers.google.com/protocol-buffers/

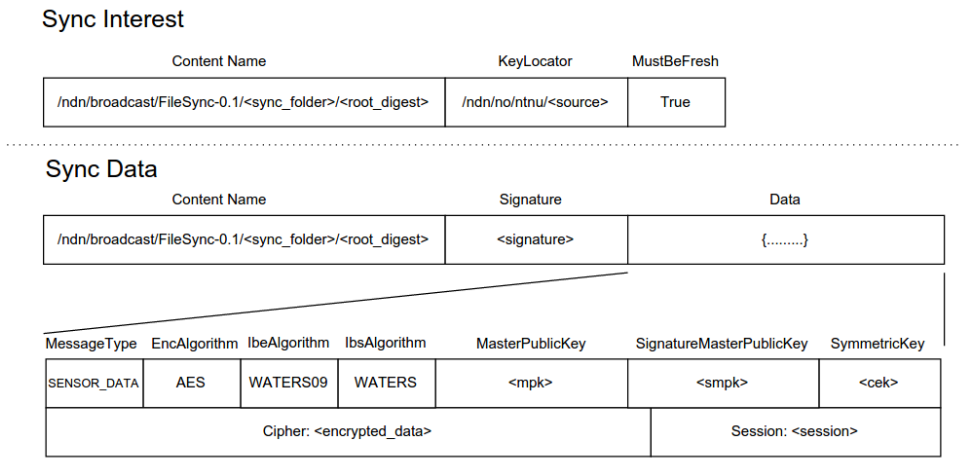


Figure 6.2: Sync Interest and Data

6.4 Health Sensor System - Implementation

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

- Device code [Mø15, device.py]
- PKG code [Mø15, publicKeyGenerator.py]
- IdentityBasedCrypto code [Mø15, identityBasedCrypto.py]
- Main code [Mø15, application.py]

figure of devices and what code/-class they run

6.4.1 Access Control

In 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.

6.4.2 Packet Design

The packet format is designed with Google Protocol Buffers. Initialization packets have the structure presented in Figure 6.3. Initially the idea was to have the *Temporary Master Public Key (TMPK)* 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 TMPK is approximately 2KB and was appended to the Content Name in the Interest, the Data response off course had to have the same Content Name, hence 2KB overhead in the Name. The TMPK 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 6.4. The code can be reviewed i [Mø15, messageBuf.proto].

Init Interest - The initialization Interest seen in Figure 6.3 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 our trust model we 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 the the initialization Interest contains data with a structure defined in [Mø15, messageBuf.proto] and illustrated in Figure 6.3.

Sensor Interest - As in the initialization Interest the KeyLocator field is used to define the requesters Name. Figure 6.4

Sensor Data - The Data response to the Sensor Interest uses the same structure as the initialization Data. It is illustrated in Figure 6.4

6.4.3 Running the Code

First the NFD must be started Listing 6.1 on each device, 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 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.

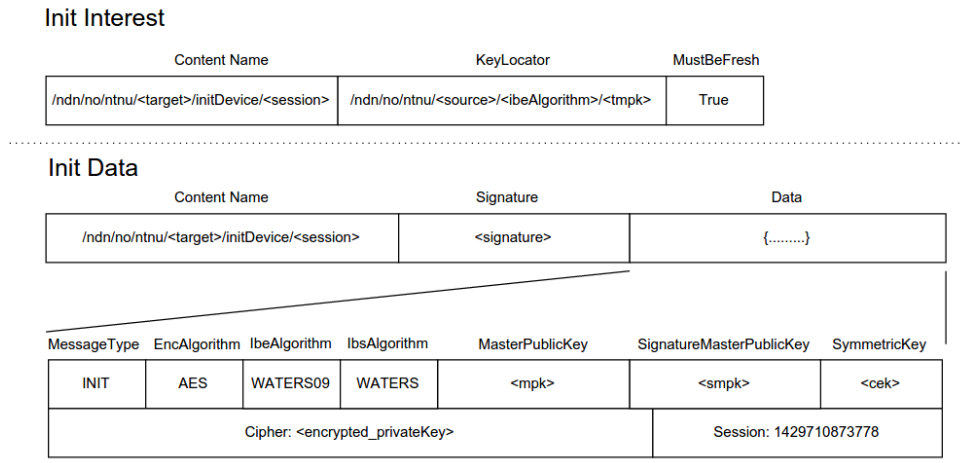


Figure 6.3: Initialization Interest and Data

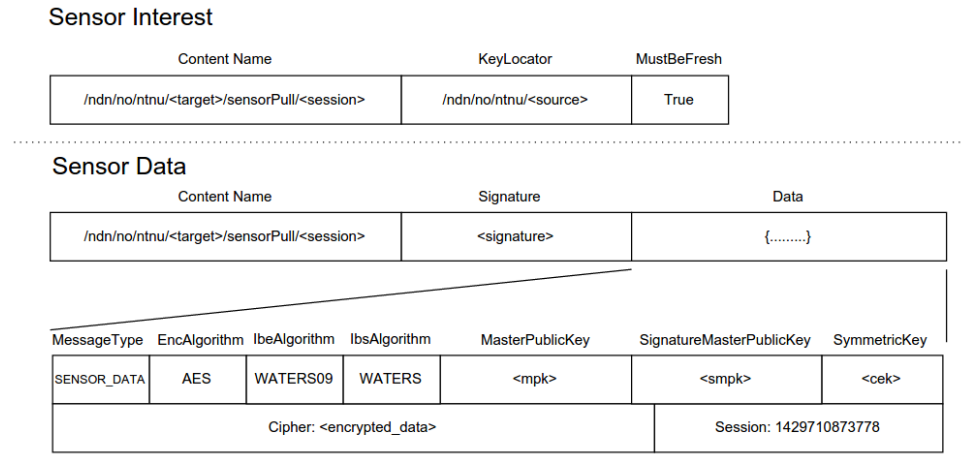


Figure 6.4: Sensor Interest and Data

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: NFD Start

```
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.

6.5 Testing

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

6.5.1 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.

6.5.2 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.

Data	Scheme	Size
Content Encryption Key (CEK)	Random	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.

6.5.3 Performance

To be able to evaluate if IBC is applicable for devices with small computation power and limited battery, it has to perform somewhat in the range of regular asymmetric encryption (read RSA), and signing. In Table 6.3 the results from running different cryptographic methods on the computers listed in Table 6.1 are presented.

The initialization protocol described in subsection 5.2.2 and the data pull protocol described in subsection 5.2.3 is tested on the computers listed in Table 6.1. The results are presented in Table 6.4.

The HSS is tested on two of the computers in Table 6.1. The topology is shown in Figure 6.5.

6.6 NDN node 23 - NTNU

Node 24 in NDN testbed

Method	Scheme	C1	C2	C3
IBE PKG Keypair generation	Waters09	0.09965	0.02709	0.03608
IBE Encrypting CEK	Waters09	0.05265	0.18915	0.02486
IBE Secret Key (SK) generation	Waters09	0.05614	0.01786	0.02327
Encrypting SK	AES	0.00013	0.00010	0.00015
IBS PKG Keypair generation	Waters	0.09755	0.02715	0.03502
IBS Secret Key (SSK) generation	Waters	0.00976	0.00287	0.00372
IBS Sign	Waters	0.00990	0.00288	0.00369
IBS Verify	Waters	0.00758	0.00266	0.00432
Encrypting SSK	AES	0.00006	0.00002	0.00004
RSA (1024-bit) keypair generation	RSA	0.25427	0.11934	0.16599
RSA Encryption	RSA	0.01480	0.00440	0.00652
RSA Decryption	RSA	0.01472	0.00445	0.00649
RSA Sign	RSA	0.01615	0.00459	0.00669
RSA Verify	RSA	0.01572	0.00453	0.00674
ECDSA Keypair generation	ECDSA	0.00057	0.00020	0.00032
ECDSA Sign	ECDSA	0.00050	0.00021	0.00033
ECDSA Verify	ECDSA	0.00090	0.00037	0.00058

Table 6.3: Cryptographic methods time chart. The time is measured in seconds and is the mean time of 100 rounds.

Protocol	C1	C1	C3
Initialization	0.2029	0.0708	0.1163
Data Pull	0.0614	0.0310	0.0462

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

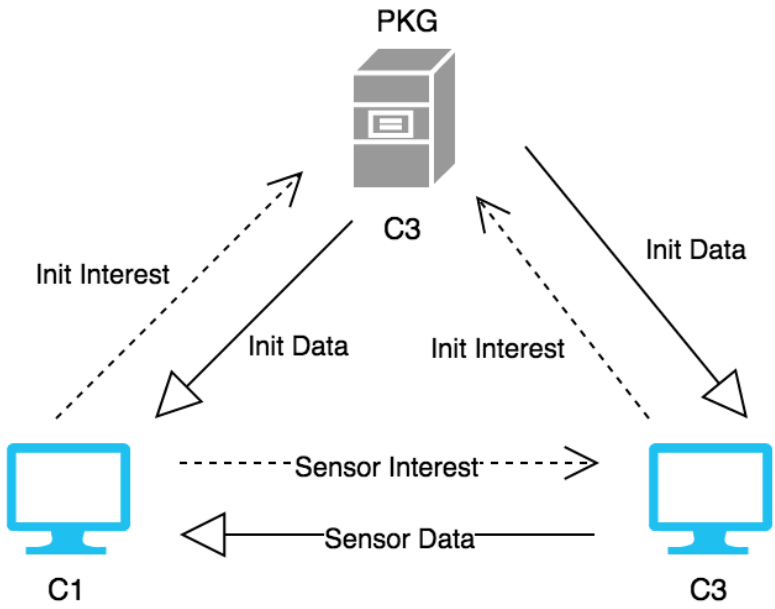


Figure 6.5: Health Sensor System implementation tested over two computer. C3 runs two nodes, i.e. the PKG and one device. C1 runs a second device.

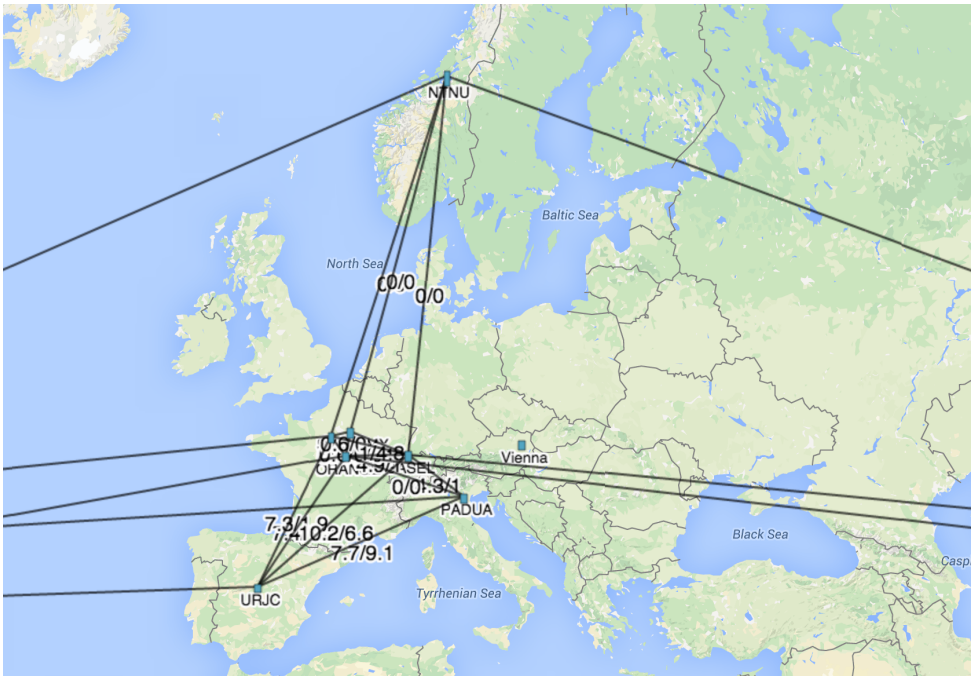


Figure 6.6: NDN Map

Chapter 7
Discussion

In this chapter the work done in conjunction to this thesis will be discussed. First I will talk about the pros and cons using IBC in NDN. Then I will discuss the HSS and possible drawbacks in the system. Scalability issues and other applicable scenarios will be mentioned.

7.1 Identity-Based Cryptography]

One suggestion has been to add a monthly timestamp [BGK12] [LV09] to the name, but the the PKG has to renew private keys for everybody each month. This solution scales very badly. With the PKGs Name Sync application, every user will be notified when a identity is revoked. There is no use for periodically checking names.

find better cite

Trusting Trusted Third Party in Identity-Based Cryptography Issues of trusting the PKG, i.e. a TTP. If the PKG is compromised by an adversary, the adversary will retrieve all private keys corresponding to all IDs that used the compromised PKG. Do every ID trust the PKG? Suspicion of MITM, where the PKG is the adversary, will always be a problem for a user. Secure channel for private key exchange.

Can authenticate Data even using insecure DNS or HTTP. There is only one linkage between the Name and the Content, and if one have the right MPK, there is no doubt where the data originates from and that it is not altered. In RSA public key cryptography we have to find the key related to the signature. In worst case this will be equivalent of retrieving the MPK each time, which is not likely. Or the MPK can be appended to the message.

7.2 Health Sensor System

Not tested on 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.

In subsection 6.5.3 we can see that IBC is performing better than regular asymmetric cryptography.

Encrypting with same symmetric key for each set of data, limits the encryption computation for the device if several devices requests the same data. Using a unique key for each time data is requested is more secure and can be used for less confidential content.

Adressing is dealt with one place in the architecture compared to an equivalent system not using NDN.

7.3 Sync

The sync application makes it possible for users to know who has a valid public key within the PKGs domain. One drawback with the IDSync solution is that 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, it has to rely on the same principle to be able to know if the belonging ID is still valid.

7.3.1 Attacks

In [SA99] Frank Stajano and Ross Anderson mentions possible DoS attacks, such as radio jamming and battery exhaustion. All applications that are relying on some sort of crucial information derived from a Sync application section 3.2 are vulnerable to DoS.

7.4 Other Use Cases

1. Home Automation Systems
2. Organization, internal communication
3. Building Automation System (BAS)
4. Building Management System (BMS)

7.5 Scalability

IDSync can be an issue, as the list that is being distributed is growing linearly with the number of participants in the trust domain.

Chapter 8
Conclusion and Future Work

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

8.1 Conslusion

In this thesis I have developed a Health Sensor System in Python with Identity-Based Cryptography used for signing and verification, encryption and decryption. The system is build over the new network protocol Named Data Networking.

I have suggested how a secure system can be implemented.

The system is tested to see how the suggested protocols for initialization and data pull performs with IBC. This work is a attempt to show how applicable NDN is for IoT and to design secure protocols for local device networks.

8.2 Future Work

Integrate the FSM in to the HSS.

Test solution on relevant sensors and devices.

Implement a full worthy IBC solution in PyNDN2.

Make a scheme that uses the same MPK and MSK to do encryption/decryption and signing/verification.

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 Saadallahy, 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 PaulAbraham, 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 CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 443–459, 2004.

[Ben07] LYNN Ben. *On the implementation of pairing-based cryptosystems*. PhD thesis, PhD thesis, Stanford University, 2007. https://crypto. stanford. edu/pbc/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.

[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.

[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.

[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-datazation 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].

[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.

[GRS05] Henri Gilbert, Matthew J. B. Robshaw, and Hervé Sibert. An active attack against HB+ - A provably secure lightweight authentication protocol. *IACR Cryptology ePrint Archive*, 2005:237, 2005.

[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.

[JW05] Ari Juels and Stephen A. Weis. Authenticating pervasive devices with human protocols. In *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, pages 293–308, 2005.

[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/master-thesis-work, 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.

[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.

[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.

[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.

[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.

[Wei05] Stephen A. Weis. Security parallels between people and pervasive devices. In *3rd IEEE Conference on Pervasive Computing and Communications Workshops (PerCom 2005 Workshops), 8-12 March 2005, Kauai Island, HI, USA*, pages 105–109, 2005.

[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 A Code

A.1 Scyther Security Analysis

```
1 hashfunction SHA256;
2 usertype ContentEncryptionKey, Content;
3
4 const pk: Function;
5 secret sk: Function;
6 inversekeys (pk,sk);
7
8
9 protocol HSSDataPull (PKG, D1, D2)
10 {
11
12   role PKG
13   {
14     #receive Interest D1
15     #send Data D1
16
17     #receive Interest D2
18     #send Data D2
19   }
20
21   role D1
22   {
23     # send Interest PKG
24     # reveive Data PKG
25
26     var Ra: Nonce;
27     fresh Rb: Nonce;
28
29     # send Interest D2
30     send_1(D1, D2,  Rb, D1,  {SHA256(Rb, D1)}sk(D1) );
```

```
31
32      # reveive Data D2
33      var CEK: ContentEncryptionKey;
34      var Data: Content;
35      recv_2(D2, D1,        Rb,    {D2, CEK}pk(D1),      { Data }k(D2
            ,D1)   ,   {SHA256( Rb,      { {CEK}pk(D1) }sk(D2),      {
            Data }k(D2,D1) )}sk(D2));
36
37      claim_D11(D1, Alive);
38      claim_D12(D1, Secret, CEK);
39      claim_D13(D1, Secret, Data);
40      claim_D14(D1, Weakagree);
41      claim_D15(D1, Niagree);
42      claim_D16(D1, Nisynch);
43    }
44
45    role D2
46    {
47      # send Interest PKG
48      # reveive Data PKG
49
50      fresh Ra: Nonce;
51      var Rb: Nonce;
52
53      # receive Interest D1
54      recv_1(D1, D2,    Rb, D1,   {SHA256(Rb, D1)}sk(D1) );
55
56      # send Data D1
57      fresh CEK: ContentEncryptionKey;
58      fresh Data: Content;
59      send_2(D2, D1,        Rb,      {D2, CEK}pk(D1) ,      { Data }k
            (D2,D1) ,   {SHA256( Rb,      { {CEK}pk(D1) }sk(D2),      {
             Data }k(D2,D1) )}sk(D2));
60
61      claim_D27(D2, Alive);
62      claim_D28(D2, Secret, CEK);
63      claim_D29(D2, Secret, Data);
64      claim_D210(D2, Weakagree);
65      claim_D211(D2, Niagree);
66      #claim_D212(D2, Nisynch);
67    }
68  }
```