NDNHealth:
A Secure mHealth Infrastructure over Named Data Networking

Saurab Dulal {sdulal@memphis.edu}
PhD Student, Computer Science
The University of Memphis

Committee
Dr. Lan Wang (Committee Chair, UofM)
Dr. Christos Papadopoulos (UofM)
Dr. Kan Yang (UofM)
Dr. Lixia Zhang (UCLA)
### Publications

**Dulal, Saurab**, Tianyuan Yu, Siqi Liu, Adam Robert Thieme, Lixia Zhang, and Lan Wang. "Enhancing NAC-ABE to Support Access Control for mHealth Applications and Beyond" (*under review*).

**Dulal, Saurab**, and Lan Wang. “Reining in Redundant Traffic through Adaptive Duplicate Suppression in Multi-Access NDN Networks” (*under review*)


Outline

- Introduction/Background
- Problem Formulation
- Related Work in mHealth Systems
- NDNHealth
- Adaptive Duplicate Suppression
- Conclusion
Mobile Health (mHealth)

- Wearable devices are growing rapidly
- mHealth data produced by these devices used for
  - Diagnosis, therapeutics, real-time medical interventions
  - Emergency management
  - Detect eating disorder
  - Detect smoking episode
  - Improve mental health/lifestyle
  - Self monitoring diet/physical activities
  - Contact tracing

![Wearable Device Shipments, Worldwide](chart)

> twice

5 years
Challenges in Collecting and Sharing mHealth Data

- Privacy and legal concerns
- Highly diverse data
- High frequency data generation (1hz to 128hz)

Designing a real-time data distribution system that maintains security and privacy for high-rate, multi-source data is a complex task [9]
Outline

- Introduction/Background
- **Problem Formulation**
- Related Work in mHealth Systems
- NDNHealth
- Adaptive Duplicate Suppression
- Conclusion
Problem Formulation

Alice wants to share her mHealth data with different consumers (doctors, instructors) based on the context (location based → {home, gym}, time based → time)

- Share blood glucose data in real-time generated at home
- Share heart-rate data generated at gym and after Apr 7, 2023
- Share blood glucose and heart-rate data generated at work

- Share mHealth data to doctors
  - real time health monitoring
  - emergency management
  - proactive healthcare

- Share mHealth data to trainers
  - monitor physical activity

- Share mHealth data to researchers
  - detecting eating disorder
  - smoking episode
  - stress

Alice (data producer)

(data consumers)
Research Questions

1. How to realize data distribution system that can support large-scale and real-time mHealth data

2. How to achieve utmost Data security
   - trust between different parties (consumer, producer, operator)
   - fine grained contextual access control on owner defined policies
Outline

- Introduction/Background
- Problem Formulation
- **Related Work in mHealth Systems**
- NDNHealth
- Adaptive Duplicate Suppression
- Conclusion
Limitation in the Existing mHealth Systems

MD2K’s [1] mHealth research cyberinfrastructure

- Cannot receive real-time data from study participants
- Limited access control

Manually identify authorized data

Encrypt and store data in flash drive

Mail the flash drive

→ Slow process, high overhead, and subject to human error ←
# Limitations in the current State-of-the-Art

<table>
<thead>
<tr>
<th>mHealth System</th>
<th>High-Frequency Data</th>
<th>Real-time Data Distribution</th>
<th>Access Control</th>
<th>Contextual Access Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Azure FHIR</td>
<td>N</td>
<td>Y*</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>NIMA</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>RADAR</td>
<td>N</td>
<td>Y*</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Adaptive MapReduce</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

* partially supported

**Contextual Access Control:** Data sharing based on the context (location, activity, time) of the data, e.g. Alice sharing her share blood sugar, heart rate data with her doctor when at home.
Use of NDN in NDNFit [2]

Named Data Networking (NDN) is a data-centric Internet architecture

- Name and secure the data, not the hosts!
- Content objects are fetched using Interest and Data packets

```
Sign, encrypt and publish the data
/org/mhealth/alice/DATA/fitness
/physical_activity/time_location/<timestamp>
```

```
Data
- name: /org/mhealth/Alice/DATA/fitness/
- Content: encrypted bits (<activity>)
- signature: digital signature by Alice
```

```
Interest: /edu/memphis/alice/heartrate/1680822655
```

mHealth Applications

Data Processing Units

Data Storage Units

Data Visualization Units

NDNFit: High Level System Overview
Limitations of NDNFit

- Use case for named based access control (NAC)
- Mostly focused on tracking and sharing personal fitness activity
- Lack of support for context-based sharing
Outline

- Introduction/Background
- Problem Formulation
- Related Work in mHealth Systems
- NDNHealth
- Adaptive Duplicate Suppression
- Conclusion
NDNHealth System Overview

Sign, encrypt, and publish heart rate data:
name: /edu/memphis/alice/heartrate/1680822655

NDNHealth is an enhancement of mGuard (modular pub-sub-API, data collect support)
Outline

● Introduction/Background
● Problem Formulation
● Related Work in mHealth Systems
● NDNHealth
  ○ Naming
  ○ Access Control
  ○ Data Authentication
  ○ Pub-Sub API
● Adaptive Duplicate Suppression
● Conclusion
Existing Naming

<sensor>–<namespace>–<device-name>–<attachment>
(e.g., ACCELEROMETER--org.md2k.autosenseble--AUTOSENSE_BLE--CHEST)

Data Stream Naming

- NDN hierarchical naming (figure →)
  based on semantic and physical properties.
- Naming is flexible, components can be added/removed

Application Data Naming

<stream name>/DATA/<timestamp>
e.g. /org/md2k/mperf/dd40c/phone/gyro/<revision>/DATA/1492637493876

Additionally, we designed naming for attributes
and manifest
Outline

- Introduction/Background
- Problem Formulation
- Related Work in mHealth Systems
- **NDNHealth**
  - Naming
  - **Access Control**
    - Data Authentication
    - Pub-Sub API
- Adaptive Duplicate Suppression
- Conclusion
Access Control Policies

- **Goal:** limit data access to only authorized users based on terms of use and privacy risks
- **Customized Access Policies** by Data Owners for Consumers

started with policies:
firewall rules and
database queries

md2k policy

NDNHealth
Access Control Policy Structure

Example Policy A
```
policy-id          2
requesters-names  /org/mhealth/bob
attribute-filters
{                  
  allow           
  {              
    <attribute 1>
    <attribute 2>
  }                 
  deny           
  {              
    /org/mhealth/alice
  }
}
```

Example Policy B
```
policy-id          3
requesters-names  /org/mhealth/saurab
attribute-filters
{                  
  allow           
  {              
    /org/mhealth/alice
    /org/mhealth/ATTRIBUTE/location/home
    /org/mhealth/ATTRIBUTE/location/gym
    /org/mhealth/ATTRIBUTE/date > 20210901
  }
}
```
Access Control: Policy Support

How to provide fine grained contextual access control?

- Named based Access Control with Attribute Based Encryption (NAC-ABE) [3]
  - Enforce access control policies on NDN names
  - Fine grained contextual access control
- Encrypt and sign the data at production
  - Ensure confidentiality, authenticity, and integrity
- Automate key distribution

High level overview: CP-ABE Based NAC
Limitations of Existing Solution

- Existing solutions $[3][4][5]$ were based on CP-ABE
- CP-ABE isn’t best fit for mHealth system
  - Requires access policy during encryption
  - mHealth system collects huge data from multiple sources.
  - Impossible to determine $\forall$ access policies during encryption
  - Future policy changes require re-encryption

- Also, existing NAC-ABE library had
  - High overhead of Content Key generation
  - Lack of support for large key sizes
  - Naming issues and inconsistencies
  - Lack of packet validation
Key Policy-Attribute Based Encryption (KP-ABE)

- Provide fine grained access based on data attributes
- Change in the access policy doesn’t require encryption

attribute set (data centric): {gps, phone, gym, timestamp = May 26, 2023}

Bob’s access control policy

```plaintext
policy-id 2
requesters-names /org/mhealth/bob
attribute-filters
{
  allow
  {
    /org/mhealth/alice/phone
  }
}
/* share all phone data stream produced by Alice */
```

Note: we use ABE to encrypt Content Keys (symmetric keys) CKs facilitate faster encryption/decryption of larger objects
CK-Caching Producer

- Original NAC-ABE encryptor generates a new CK for every piece of data
- We implemented CK Cache Producer
  - Balances
    - Overhead of generating new CKs
    - Security risks due to compromised CKs.

CK update/configuration
- # number of data are encrypted using $\text{CK}_i$
- a specific interval has passed

![Diagram of CK-Caching Producer process]
Updated NAC-ABE Naming Scheme

e.g. /org/mhealth/aa

<aa-prefix>
PUBPARAMS
<abe-type>
<version>
( public params )

DKEY
<consumer public-key name>
<version>
( decryption key )

<segment>

<producer-prefix>
CK
<version>
<encryption attributes or policy >
<segment>
( content key )

<data suffix>
e.g. .../cgm/blood-glucose/<timestamp>
( data name )
Outline

● Introduction/Background
● Problem Formulation
● Related Work in mHealth Systems
● NDNHealth
  ○ Naming
  ○ Access Control
  ○ Data Authentication
  ○ Pub-Sub API
● Adaptive Duplicate Suppression
● Conclusion
Trust Model in NDNHealth

How to verify fetch data is authentic?
- establish trust between different entities in the system
- verify data using data signatures and the trust model

System operator (trust anchor)

certifies

mguard controller

sign
verify

/org/mhealth/controller/KEY

/configured with trust anchor

sign
verify

publisher

consumer

/org/mhealth/alice/sensor/ecg/<revision>/DATA/1680822655
/org/mhealth/alice/phone/gps/<revision>/DATA/1680822656

How to verify fetch data is authentic?

- establish trust between different entities in the system
- verify data using data signatures and the trust model
Packet Validation: Authenticity and Integrity

- Added data validation in NAC-ABE
- Helps entities authenticate the packets (Keys and Application data)

```
/org/mhealth/KEY
  Trust anchor Key

/org/mhealth/aa/KEY
  Attribute Authority Key

/org/mhealth/aa/PUBPARAMS/<>
  Public Params

/org/mhealth/aa/DKEY/<>*
  Decryption Key

/org/mhealth/alice/KEY
  Data producer Key

/org/mhealth/aa/CK/<>/ENC-BY/<>
  Content Key

/org/mhealth/alice/sensor/ecg/DATA/<timestamp>
  Applicate Data
```

NAC-ABE Internal Packet Exchange

```
{ Public Params }

NAC-ABE Producer

{ Applicate Data and CK Packet }

NAC-ABE Consumer
```

Attribute Authority

{ Public Params, DKEY }
Outline

● Introduction/Background
● Problem Formulation
● Related Work in mHealth Systems
● NDNHealth
  ○ Naming
  ○ Access Control
  ○ Data Authentication
  ○ Pub-Sub API
● Adaptive Duplicate Suppression
● Conclusion
Existing Solution

- Existing NDN Sync based pub-sub-APIs \cite{6,7} don’t support fine-grained contextual access control

- Our work is inspired by Yu et. al.’s work \cite{8}, a pub-sub-API for NDN-Lite with built-in security
  - However, it wasn’t built on Sync also doesn’t support contextual access control
Publisher Subscriber API in NDNHealth

- Builds off NDN Synchronization (i.e. PSync)
- Security is embedded inside the API
- Authorizes subscribers according to Access Control Policies
- Provides real-time notification to subscriber
- Guarantees the reception of the latest, verified data

```
publish(dataName, attributes, content)
- dataName: /org/mhealth/alice/phone/gps/DATA/<revision>/1680822656
- attributes: {/org/mhealth/ATTRIBUTES/location/home, ....}
- content: "2022-05-05 10:39:00,2022-05-05...."

subscribe/unsubscribe([dataStreams], callback)
- dataStream: /org/mhealth/alice/phone/gps
- callback: "2022-05-05 10:39:00,2022-05-05...."
```

```
sync interest (I) : /org/mhealth/sync/<subscription-list>
sync reply (D): /org/mhealth/sync
content : {/org/mhealth/alice/phone/gps/DATA/<revision>/1680822656, ....}
```
Publisher Subscriber API (cont..)

- Signing/Verifying millions of data packets using public-key crypto is computationally expensive
- Publishing individual object names through PSync lead to excessive Sync message overhead, high delays, losses

Manifest

- Naming: <data stream name>/MANIFEST/<seq-num>
  - Benefits
    - Bulk verification
    - Application Names → Sequential Names

Note:

- Use of manifest is optional
- Number of data names that goes into the manifest is configurable

---

**Data stream manifest**

name: /org/mhealth/alice/phone/gps/<revision>/MANIFEST/<seq-

content

/.../DIRECTORY/CONTENT/DATA/<timestamp(t1)>/<digest>
/.../DIRECTORY/CONTENT/DATA/<timestamp(t2)>/<digest>
/.../DIRECTORY/CONTENT/DATA/<timestamp(t3)>/<digest>
.............../DIRECTORY/CONTENT/DATA/<timestamp(tn)>/<digest>

**Fig: Manifest Data Format**
NDNHealth: Implementation

- NDNHealth is implemented in C++
- It uses the following NDN packages
  - ndn-cxx
  - NFD
  - ndn-tools
  - PSync
  - NDN Python Repo
Performance Evaluation

( Setup: Emulated using Mini-NDN and NDN testbed topology (37 nodes, 94 links) )

1. Rate experiment
   ○ Used Policy: access all the streams under /org/md2k/mperf/dd40c
   ○ Publication rate: 1-8 data points per second

→ normalized delays increase with data generation rate ←
→ cryptographic delays remain constant ←
Performance Evaluation

→ measured packet count is about 61% and 47% lower than the maximum packet count at the producer and each consumer respectively ←
→ due to interest aggregation and data caching ←
Performance Evaluation

2. Packet Size Experiment

- **Used Policy**: access only the battery stream `/org/md2k/mperf/dd40c/phone/battery`
- **Publication rate**: 300 data points per minute

Fig: Network, Processing, and Cryptographic Delays for Rate Experiments

→ the normalized delays are constant or decreasing and are much improved compared to the rate experiments ←

→ cryptographic delays increased linearly←
Outline

- Introduction/Background
- Problem Formulation
- Related Work in mHealth Systems
- NDNHealth
- **Adaptive Duplicate Suppression**
- Concussion
Background/Problem

On a multi-access link (e.g., WiFi or Ethernet), multiple nodes may:

- send the same Interest, i.e., request for the same data
- respond with the same Data packet

NDN Forwarder lacks a duplicate suppression mechanism → significant network congestion.
Existing Solutions

- Random backoff (wait) before sending Interest/Data packet [9]
  - Not adaptive to network condition

- Leader based Interest suppression [10]
  - Leader can be a bottleneck, other nodes won’t get fair chances to fetch the content
Adaptive Duplicate Suppression (ADS) Goals

- Reduce the number of redundant requests and data replies to increase overall available bandwidth (in one-hop scenario)

- Ideally the suppression should adapt to the network condition.
  - Lossless environment: only 1 interest/data of the same name by all nodes
  - Lossy environment: allow a small number of duplicates
Use # duplicate to estimate suppression time

● Record multicast Interest/data into a table
  ○ Every entry has a short lifetime
  ○ If Interest is satisfied or entry expired, remove from the table

● Before sending a packet, check if it is already in the table
  ○ If it is in the table, drop the Interest or data.
  ○ Otherwise, go to the process in the next slide

<table>
<thead>
<tr>
<th>name</th>
<th>type</th>
<th># duplicate count</th>
</tr>
</thead>
<tbody>
<tr>
<td>/mhealth/alice/ecg/1688993223</td>
<td>Interest</td>
<td>2</td>
</tr>
<tr>
<td>/mhealth/bob/ppg/1688993423</td>
<td>Data</td>
<td>1</td>
</tr>
</tbody>
</table>
High Level Design

If there is no record in the measurement table

- Wait (suppression time) before forwarding
- If another Interest/data is overheard, cancel forwarding
- The suppression time should be adaptive
  - Generally,
    - suppression time $\propto$ duplicate count
  - For lossy links, permitting a few duplicates should be ok, so the wait time can be smaller
Updating duplicate count of Interest/Data

- Exponential Weighted Moving Average (EWMA) of the duplicate Interest/data count is computed using following formula.

\[
EWMA_t = \begin{cases} 
  dc_1 & t = 1 \\
  \alpha \cdot dc_t + (1 - \alpha) \cdot EWMA_{t-1} & t > 1 
\end{cases}
\]

\[
\alpha = \text{smoothing factor}, \ t = \text{initial state}, \ dc = \text{duplicate count}
\]

- EWMA is computed every time an entry expires in the measurement table
- Granularity = name – last component
  - e.g., if name = /a/b, EWMA will be of /a
- EWMA is reset after MAX_MEASUREMENT_INACTIVE_PERIOD (e.g., 5 minutes)
Adaptive Suppression Time

**Goal:** Maintain the duplicate count below the DUPLICATE_THRESSHOLD (Dthresh)

**Phase 1:** EWMA > Dthresh AND increasing

**Phase 2:** EWMA < Dthresh AND increasing  
AND EWMA > Dthresh AND decreasing

**Phase 3a:** Initial stage, no duplicate

**Phase 3b:** EWMA < Dthresh AND decreasing

Suppression time oscillates between the phased to discover optimal operating range.

*Phases of suppression time.* The graph is drawn using the data from one of the experiments.
We compute the wait time using following formula;

\[ waitTime (w_i) = rand (0, 2 \times st_i) \]
ADS Implementation and Experimental Evaluation

- Implemented (C++, GitHub), in the current NFD @ face (link-service)
- Experiment setup
  - Emulator: Mini-NDN-Wifi
  - Consumers fetch 1MB file (Packet Size 1100-byte, 956 segments) using catchunks/putchunks over multi-access link (WiFi)
  - Ran experiments for IEEE802.11g and IEEE802.11b which supports 54Mbps and 14Mbp respectively
Evaluation: Average Packet Count

With 12 consumers and IEEE802.11g, the producer receives 54% fewer Interests and sends out 64% fewer Data packets compared to the second-best strategy (rand-15).

Similarly, each consumer receives 53% fewer Interests, 62% fewer Data and sends out 81% fewer Interests.
Evaluation: Goodput and File Transfer Time

with 12 consumers and IEEE802.11g, ADS achieves 2x to 3x better goodput than rand-15 and rand-10, and almost 8x better than no duplicate suppression.
Evaluation: Duplicate Packet Suppressed

Delayed start: at first, n-1 (all but one) consumer fetches simultaneously, after 4 seconds, n^{th} consumer start fetching

<table>
<thead>
<tr>
<th>topo</th>
<th>dup interest suppressed</th>
<th>dup data suppressed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>rand-10</td>
<td>rand-15</td>
</tr>
<tr>
<td>1p4c</td>
<td>63.90%</td>
<td>74.97%</td>
</tr>
<tr>
<td>1p6c</td>
<td>69.94%</td>
<td>80.16%</td>
</tr>
<tr>
<td>1p8c</td>
<td>67.61%</td>
<td>79.89%</td>
</tr>
<tr>
<td>1p10c</td>
<td>68.06%</td>
<td>79.40%</td>
</tr>
</tbody>
</table>

With 1 producer and 10 consumers, ADS suppressed 85.60% of duplicate data, whereas rand-10 and rand-15 suppressed 1.95% and 3.12% respectively.
Outline

● Introduction/Background
● Problem Formulation
● Related Work in mHealth Systems
● NDNHealth
● Adaptive Duplicate Suppression
● Conclusion
Conclusion

Contributions

● NDNHealth: A Secure mHealth Infrastructure over NDN
  ○ A high-level access control policy/specification
  ○ Fine grained contextual data access control
  ○ Publisher – Subscriber API with build in Security
  ○ Improved NDN ABE (NAC-ABE) library with KP-ABE, validation, segmentation and more

● Adaptive Duplicate Suppression for Multicasting in NDN
Discussion

- Developing Non-Toy NDN Application is not simple

- End-to-end Data Privacy
  - Data producers should be able to track and verify the sharing of their data.

- Access revocation
  - Revoking consumer’s access right, how and when?
Future Work

● Future work on NDNHealth
  ○ Data Collection
  ○ End-to-end privacy
  ○ User study with deployment/experiment on NDN testbed

● Future work on adaptive duplicate suppression
  ○ Determine appropriate prefix granularity for measurement
  ○ Investigate the possibility of using ML to determining the suppression time
  ○ Expand the work to multi-hop scenario
References


