CS217B Project Report: NDN Open mHealth Pilot Applications

Zhehao Wang, 404380075
zhehao@cs.ucla.edu

1 Introduction

Mobile health has emerged as both an important commercial market and a key area of Health IT, a national priority [1]. The Open mHealth team envisions that the Internet will interconnect data capture, secure storage, modeling and analytics, and user interface components to create a modular, layered framework. In [5], the Open mHealth architecture uses a data-centric hourglass model, with the "thin waist" representing the storage unit and standardized data exchange. The focus on data exchange as the backbone of the application ecosystem makes Open mHealth an excellent network environment to both drive and evaluate NDN [4].

The design work is mostly done in 2015, and this quarter's work focuses on the implementation side, with the goals being to create a demonstrable system which incorporates name based access control for data confidentiality, DPU data encapsulation, and producer mobility support in all the involved components. The first goal two goals are achieved and demonstrated in sample components, and the work on the last goal is still ongoing.

The goal of this project is to provide a working set of application components for the NDN Open mHealth (NDN omh) system, and in doing so, identify requirements for the architecture, and put new library features to test. For example, all components in the system would be customers for schematized trust [7], name based access control (NAC)[6], link objects, etc.

2 Progress this quarter

The major components and dataflow of NDN omh are shown in Figure 1. Data storage units (DSU) serve as passive repositories for user's health data, and other user information such as their certificates and associated link objects. Data processing units (DPU) are user-authorized processing units, which consume users' raw data and produce processed data. The current system uses NFN for DPU [2]. Data visualization units (DVU) are consumers in the system, and provide visualization to user's health data.

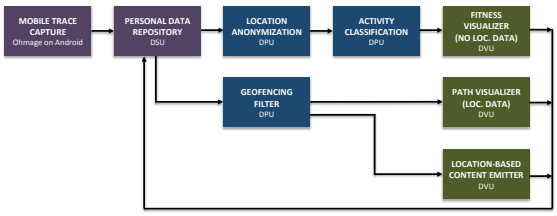

Figure 1: NDN omh components and dataflow [1]

Since the last quarter progress has been made towards the mobile data producer, mobile identity manager and trust configuration, and the DSU. Initial incorporation of NFN is done during the 2016 Hackathon, though the resulting work didn't incorporate access control at that point.

This quarter's implementation includes the following, and the following subsections introduce the implementation details worth mentioning.

- A sample producer in Python which puts encrypted user health data into DSU

- A sample authorization manager in Python which receives access requests to user's data, decides if access should be granted, and serves as a group manager for NAC. The authorization manager also publishes the group's E-key and distributes each member's encrypted D-key.
- A sample DPU in Python which requests access for user data, and consumes raw data to produce processed data on command interest.
- A browser DVU in JavaScript which requests access for user data, decrypts user data and visualize, initiates DPU processing, and visualize all the data names in the network.

2.1 Sample producer

This producer is a simplified version of NDNFit Android application. It produces random time-location data for a pre-configured user, puts the produced data into its own memory content cache, and initiates DSU insertion, which is an NDN repository running repo-ng. The producer can be configured to produce encrypted data, in which case it'll issue interest for the E-keys of groups whose name is a prefix of the produced data name, verify the received E-keys, and publish the C-keys encrypted with each of the E-keys to the DSU.

The producer is built with PyNDN, whose NAC support is a direct port from Yingdi's version in C++. During the implementation of producer (in Hackathon) we identified an issue with both versions of NAC library, which wrongfully rounds the produced data's timestamp and causes the producer's finest time granularity to be every hour. This seems to be a simple implementation oversight, and was quickly fixed during the Hackathon. A separate issue is the exclusion mechanism with which the library fetches E-keys: whether the exact hour point is included or excluded. For our demonstration the behavior is changed to inclusive rather than the exclusive version in the library, so that a E-key for an hour period can cover the data produced at the exact starting hour. This issue's still pending in the library implementation.

The incorporation of NAC into the Android producer is still being tested.

2.2 Sample authorization manager

This authorization manager derives from the group manager in NAC, which provides interfaces for adding a member certificate to the group, and generating all the group keys (one group E-key and all D-keys for all members of the group). Access request and member certificate fetching are considered application-specific logic, and are out of the scope of the NAC library. This group manager receives membership request interests, which contain the name of the requesting identity's certificate. It then asks for the cetificate and grants access to anyone verifiable from a pre-installed anchor.

During the implementation of the group manager, we found a potential issue with the library's implementation of *groupManager.getGroupKey*, the interface which returns the E-key of the group and all current member's D-keys. The current implementation regenerates all key pairs upon each call, which in our case would invalidate the C-key encrypted with a previous E-key, and the producer has to be aware of the new E-key each time this function is called (which needs to happen every time a new member's added, in order to publish that member's encrypted D-key). For our demonstration, the behavior of this function is changed to only generate E-key once, and upon each call only generate D-keys for members whose D-keys don't previously exist. This potential issue's still pending in the library implementation.

We expect the actual authorization manager to also be an Android application running on the same device as the producer application, and the authorization manager would contain multiple group managers and an access request interface, which upon receiving requests, would allow user to choose who to grant access to (instead of relying on a pre-installed trust schema), in which group, and for how long.

2.3 Sample DPU

Our sample DPU operates under its organization namespace: it receives command interests in its organization namespace, and wants to produce data available in the user namespace. To achieve this the DPU produces data under the organization names-

pace, so that the command interest is satisfiable, and in the data is another data packet under the user namespace, so that the receiving end, or the DPU itself can put it in DSU and make it available under the user namespace; both outer and inner data are signed by DPU's key. For our demo, calling the function with the following outer name

```
/ndn/edu/basel/dpu/bounding_box
(<userid>, <timestamp>, <inner_data_name>)
```

will generate the inner name that looks like the following

```
/org/openmhealth/<userid>/data/fitness
/physical/processed_result
/bounding_box/<timestamp>
```

in the parameters `<userid>` and `<timestamp>` are mandatory, and `<inner_data_name>` is optional if the caller wants to further customize the generated inner data name.

In addition to doing encapsulation, the DPU also serves as both a NAC consumer and producer. As a consumer it first requests access, then fetch the catalog, and using names in the catalog to ask for data and the correct credentials to decrypt the C-key. As a producer it respects the data owner's group setup by requesting the same E-keys thus making sure the same groups have access to the data.

The implementation of DPU shares the same potential library issues with the sample producer, with the same revisions on the library as described previously, we were able to implement a DPU that provides a simple bounding box function.

Ideally we want to use an NFN component as the DPU, but the incorporation of NAC into NFN is still being tested.

2.4 DVU

The DVU runs in browser and provides the options of requesting user data access, fetching a user's raw or processed data, initiating DPU processing with a given name, and show all data names in the network (organized as a tree). The implementation uses ndn-js's port of NAC, which did not require any direct change since the consumer implementation in NAC is working as expected.

2.5 Demo setup and names involved

This section describes the process and names involved in the initial demo of the system [1]. Figure 2 introduces the sequence of events in the demo, featuring the access granting and raw data consumption part.

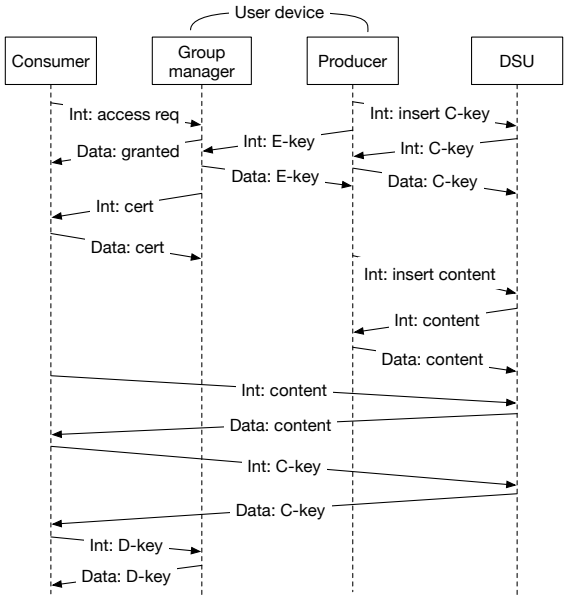

Figure 2: NAC demo workflow

And the names involved include:

- User namespace (group name):

  ```
  /org/omh/zhehao
  ```

- Encrypted content name:

  ```
  /org/omh/zhehao/SAMPLE/fitness/physical
  /time_location/20160320T080058
  /FOR
  /org/omh/zhehao/SAMPLE/fitness/physical
  /time_location/C-KEY/20160320T080000
  ```

- Catalog name:

  ```
  /org/omh/zhehao/data/fitness/physical
  /time_location/catalog/20160320T080000
  /%FD%01
  ```

[1] Which can be found at: https://www.youtube.com/watch?v=3l2w30rZqdk; and the corresponding implementation is https://github.com/zhehaowang/sample-omh-dpu

- DVU identity name:

  `/org/omh/dvu-browser/`

- DVU access request interest:

  `/org/omh/zhehao/read_access_request`
  `/org/omh/dvu-browser/KEY/ksk-1464919278`
  `/ID-CERT/%FD%00%00%01U%13%FEC%BA/`
  `%FD%00%00%01U%14t%E7%F9`

- Encrypted C-key:

  `/org/omh/zhehao/SAMPLE/fitness/physical`
  `/time_location/C-KEY/20160320T080000`
  `/FOR`
  `/org/omh/zhehao/READ/fitness/E-KEY`
  `/20160320T080000/20160320T100000`

- Group E-key name:

  `/org/omh/zhehao/READ/fitness/E-KEY`
  `/20160320T080000/20160320T100000`

- Consumer D-key name:

  `/org/omh/zhehao/READ/fitness/D-KEY`
  `/20160320T080000/20160320T100000`
  `/FOR`
  `/org/openmhealth/dvu-browser`
  `/ksk-1464919278`

- DPU identity name:

  `/ndn/edu/basel/dpu/`

- And DPU function call name and produced data names are given in the previous section.

3 Conclusion

What we achieved can be summarized as providing sample implementations for all the involved components, featuring name based access control and DPU data encapsulation; the details are introduced in previous section. This section discusses what we learnt and the immediate future work for the project this quarter.

3.1 What we learnt

What we've learnt can be summarized as the following bullets:

- Implementation experience: this project involves a considerable amount of implementation in multiple languages (mostly done in Python and JavaScript) and platforms (components are running on OSX, Ubuntu, Android and in browser), and the resulting implementation and integrationt testing experience is valuable. (The process of looking inside the library to debug is helpful, too, in understanding how exactly everything works)

- NAC library usability: this application's also the first one that fully features the NAC library, whose debugging, for now, requires pretty detailed review of the code. We did identify a few potential issues, and hopefully could help improve the usability of NAC library, and provide a working example for other applications wishing to feature data confidentiality and access control.

- Writing experience: earlier this quarter we did a short paper submission for ICN, which could serve as an initial documentation of the system design. Though the writing quality as a paper ends up being subpar, the process of finding selling points in the project and organizing the ideas to make them presentable is still helpful.

- The process of designing, building a system, then identifying the problems and making improvements: this is something we are hoping to learn from building the NDN Open mHealth system, though we can't state that we've learnt a lot in this aspect, as the work this quarter can still be categorized as the initial stage of building the prototype. We are hoping to learn more from seeing the system in action, and evaluate/improve in the next iteration.

And as a side note, we did not run into many problems as other groups did when testing on the testbed, or installing dependencies, most likely because of the familiarity with existing tools, libraries and how they work.

4 Future work

Implementation and deployment wise, the immediate future work includes the following.

- Incorporating with real components: as discussed in the previous section, most of the components serve as sample implementations rather than real components to be demonstrated with the system. Some effort should go into actually building the components as we imagined (for example, the authorization manager in Android; and incorporating schematized trust as in the design document).
- System deployment, evaluation and better visualization. With more initial implementations in place, we can move on with actual deployment on testbed, and decide how to best evaluate and demonstrate the system. The current idea's to visualize the interest and data exchange between different components, however, more thought needs to go into designing the demo so that it's clear and helpful, or better yet, to be able to evaluate with the IP implementation from certain aspects.

References

[1] NDN Open mHealth Tech report, `https://github.com/remap/ndn-netenv-techreports/tree/master/Open_mHealth`.

[2] NFN website, `http://www.named-function.net/`.

[3] Screen recording of omh components demo, `https://www.youtube.com/watch?v=3l2w30rZqdk`.

[4] Fia-np: Collaborative research: Named data networking next phase (ndn-np) proposal. Tech. rep., Technical Report NDN-0026, NDN Project, 2014.

[5] ESTRIN, D., AND SIM, I. Open mhealth architecture: an engine for health care innovation. *Science 330*, 6005 (2010), 759–760.

[6] YINGDI YU, ALEXANDER AFANASYEV, L. Z. Name-based access control. Tech. rep., Technical Report NDN-0034, NDN Project, 2015.

[7] YU, Y., AFANASYEV, A., CLARK, D., CLAFFY, K., JACOBSON, V., AND ZHANG, L. Schematizing trust in named data networking. In *Proceedings of the 2Nd International Conference on Information-Centric Networking* (New York, NY, USA, 2015), ICN '15, ACM, pp. 177–186.