Secure Bootstrapping and Access Control in NDN-based Smart Home Systems

Lei Pi, Lan Wang
{lpi, lanwang}@memphis.edu
University of Memphis

***Abstract*—Ensuring homeowners' privacy and security is of vital importance in smart home systems. We adopt a data-centric approach to develop a smart home system over Named Data Networking. We address two specific problems: (a) establishing a trust relationship between a new device and the system, and (b) ensuring only authorized users can access data in the system. Our implementation allows a homeowner to use a smartphone to add new devices and control access at a fine granularity.**

I. Introduction

Smart Home systems utilize sensor-collected data to provide intelligent services to homeowners. However, these systems pose potential security risks. Once an attacker gains access to a smart home system, he/she can not only remotely monitor and manipulate it, but also perform physical intrusions. These risks are amplified when a homeowner is not an expert in configuring such systems. Industrial solutions such as Google Threads, Samsung SmartThings, and ZigBee Home Automation rely on remote cloud servers or build on complex networking protocols. Moreover, they lack strong mechanisms to protect homeowners from unauthorized data production and access.

This poster shows how we use Named Data Networking (NDN) [1] to build a simple cloud-independent smart home system focusing on securing data production and access. Our solution includes a bootstrapping protocol that is secure against replay and man-in-the-middle attacks, an implementation of Name-based Access Control (NAC) [2] for smart homes, and a programming library that eases the effort to adopt our designs on macOS, Linux and Android platforms.

II. Background on Named Data Networking

NDN is a promising realization of the Information-Centric Networking paradigm. It provides a data-centric model which decouples the production and consumption of data, provides in-network caching, and secures data directly. In NDN, an application fetches data identified by a name directly by sending an Interest packet and receiving a Data packet. Each NDN node forwards an Interest packet according to its name to other nodes until it reaches a producer or a node holding a cached copy. Each Data packet is signed by its producer's public key. With schematized trust [3], a developer can easily automate trust rules to check data integrity and authenticity.

III. Design and Implementation

An IoT device's life cycle in NDN has the following phases. First, the device performs necessary hardware checks, loads the OS, and starts network configurations. Second, it gains physical connectivity with other devices on the same network. Third, the device initiates trust bootstrapping with the homeowner to learn the trust anchor and establish an identity of itself by retrieving a certificate signed by the owner. Fourth, the device produces data for the homeowner and other authorized users to access. Lastly, the device may be reset from time to time and finally be decommissioned. Our work focuses on the third and fourth phases.

A. Naming Hierarchy and Trust Model

We designed a hierarchical namespace for a smart home system starting with a "`local-home`" component (Figure 1). The "`Key`" branch contains the key names of the owner and other entities. The "`bootstrap`" branch is for naming data packets used in the bootstrapping process. The "`samples`" and "`read`" branches are required by Name-based Access Control (NAC) [2], where the former is for naming produced data and the latter is for naming production/consumption credentials.

An NDN identity is the binding of a namespace and a public key for a specific period. It is implemented in the form of an NDN *Certificate* and can be used to authenticate data produced by a real-world entity. An NDN trust model defines which identities act as the *trust anchors*, i.e., self-signed identities that are trusted by the system by default. It also defines rules to determine which identity can prove the authenticity of another identity by signing the latter's certificate.

In the smart home system, the homeowner's self-signed certificate is the trust anchor and it is bound to the "`local-home`" namespace. If an identity under the "`local-home`" namespace is signed by the trust anchor, it is valid. Otherwise, an identity is valid only if 1) its namespace is a sub-namespace of its signer, and 2) its signer is a valid identity.

B. Bootstrapping Trust

When a new device joins the smart home network for the first time, it needs to learn the trust anchor and its namespace for data production. The trust anchor should learn the device's public key and generate a certificate for the device which contains the device's public key, namespace, and the trust anchor's signature. Upon a successful trust bootstrapping, the device should gain confidence that it joined a trusted smart home environment with the homeowner's permission, and its published data will be trusted over the network. Then it may fetch runtime configurations or begin publishing data. The

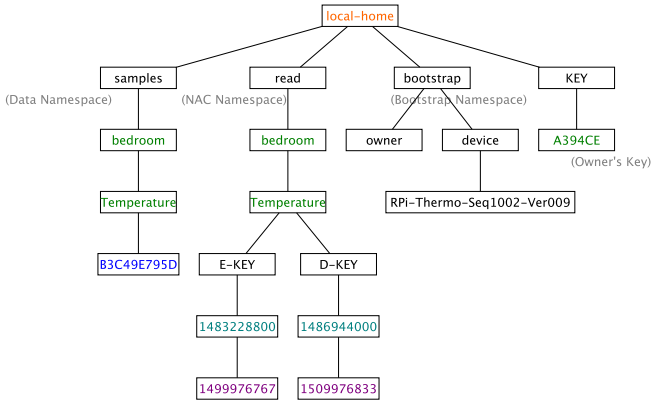


Fig. 1: Namespace Design

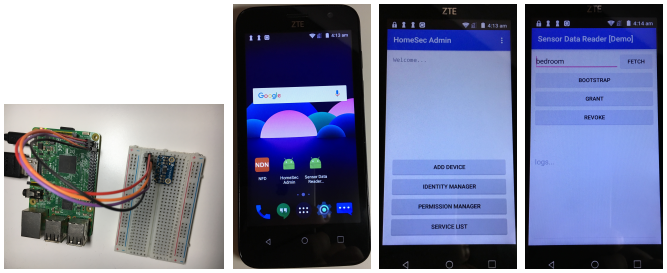
homeowner's controller app (acting as the trust anchor) should gain confidence that the new device is what it claims to be. It is vital that during this process each party can check message authenticity and detect replay and man-in-the-middle attacks.

The bootstrapping protocol between a new device D and the homeowner's controller app C works as follows.

1) D initiates the process by sending an interest "`/local-home/bootstrap/owner/<device-id>/`R_0`/HMAC(<device-id>|`R_0`, K)`". K is a pre-shared secret between the device D and owner's controller C. *HMAC* is a keyed-hash message authentication code function. R_0 is a random number generated by D.
2) Once C verifies the HMAC in D's bootstrapping interest, it sends back the trust anchor's self-signed certificate $Cert_o$ and HMAC$(Cert_o|R_0, K)$ in a data packet. D verifies the message by recalculating the HMAC.
3) C sends an interest "`/local-home/bootstrap/device/<device-id>/`R_0`/`R_1`/HMAC(<device-id>|`R_0`|`R_1`, K)`". R_1 is a random number generated by C.
4) Once device D verifies the HMAC in the received interest, it sends its public key P_D, R_1, and HMAC$(P_D|R_1, K)$ in a data packet.
5) After C verifies P_D using R_1 and K, the owner issues a certificate $Cert_D$ by signing P_D (the certification can be done in C or a separate certificate issuing app). D then fetches its certificate by sending the interest "`/local-home/bootstrap/cert/<device-id>`".

C. Name-based Access Control

Fine-grained access control to smart home data helps defend against privacy leaks and intrusions [4]. Name-based Access Control (NAC) is a form of encryption-based access control using keys associated with specific name scopes [2]. It requires data to be encrypted upon production, and it authorizes production and consumption by distributing encryption and decryption keys. We applied NAC to the smart home system and implemented it on x64 and ARMv8 platforms by developing cross-platform libraries and dedicated apps for macOS, Linux, and Android systems. Our implementation consists of two parts: the common smart home node API library and applications. The applications include NAC manager daemon



(a) (b) (c) (d)

Fig. 2: Smart Sensor and Android Apps

on macOS, owner's controller and data consumer apps on Android, and data producers in Ubuntu on Raspberry Pi. Figure 2 shows (a) the Raspberry Pi with a connected digital thermometer (MCP9808), (b) the owner's Android phone with the NDN Forwarding Daemon, (c) the owner's controller app for managing access to home devices, and (d) the data consumer app for accessing data.

IV. Related Work

In [5], Shang et. al. discussed challenges to realize the IoT vision in a host-centric network architecture and explains how NDN can address the root causes of these challenges. Bannis and Burke [6] introduced a gateway-based architecture to build a secure integrated home network over NDN. It requires a synchronized clock between devices before bootstrapping. *NDN-Flow* [7] is an application focusing on trust management and service rendezvous. Compagno et. al. [8] have different assumptions and objectives for the on-boarding process which lead to a more complex protocol.

V. Future Work

We plan to compare the security and performance of our solution with other proposals. We will also use the same NAC framework to securely control home devices, e.g., turning on and off a video camera, over a home network.

References

[1] L. Zhang, A. Afanasyev, J. Burke, V. Jacobson, P. Crowley, C. Papadopoulos, L. Wang, B. Zhang *et al.*, "Named data networking," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, pp. 66–73, 2014.

[2] Y. Yu, A. Afanasyev, and L. Zhang, "Name-based access control," *NDN Technical Report NDN-0034*, 2015.

[3] Y. Yu, A. Afanasyev, D. Clark, V. Jacobson, L. Zhang *et al.*, "Schematizing trust in named data networking," in *Proceedings of ACM ICN*, 2015, pp. 177–186.

[4] E. Fernandes, J. Jung, and A. Prakash, "Security analysis of emerging smart home applications," in *IEEE Symposium on Security and Privacy (SP)*, 2016, pp. 636–654.

[5] W. Shang, A. Bannis, T. Liang, Z. Wang, Y. Yu, A. Afanasyev, J. Thompson, J. Burke, B. Zhang, and L. Zhang, "Named data networking of things," in *IEEE International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2016, pp. 117–128.

[6] A. Bannis and J. A. Burke, "Creating a secure, integrated home network of things with named data networking," 2015.

[7] W. Shang, Z. Wang, A. Afanasyev, J. Burke, and L. Zhang, "Breaking out of the cloud: local trust management and rendezvous in named data networking of things," in *IEEE International Conference on Internet-of-Things Design and Implementation (IoTDI)*, 2017, pp. 3–14.

[8] A. Compagno, M. Conti, and R. Droms, "Onboardicng: A secure protocol for on-boarding iot devices in icn," in *Proceedings of the 3rd ACM Conference on Information-Centric Networking*, 2016, pp. 166–175.