Secure Sensing over Named Data Networking (Extended Version)

Jeff Burke
University of California, Los Angeles
jburke@remap.ucla.edu

Paolo Gasti
New York Institute of Technology
pgasti@nyit.edu

Naveen Nathan
Gene Tsudik
University of California, Irvine
{nnathan,gts}@ics.uci.edu

***Abstract*—The anticipated proliferation of smart devices, the "Internet of Things" (IoT), is one of the motivations for some large-scale research efforts aiming to design a new Internet architecture. One such effort is Named-Data Networking (NDN) – a "future internet architecture" research project in the Information-Centric Networking (ICN) area that emphasizes efficient, scalable and secure data distribution through a shift from the host-based addressing of IP to data-centric addressing. Because of its focus on data distribution, NDN has been assumed to be poorly suited for other networking scenarios.**

We address efficient and secure sensing over NDN, motivated by the convergence of the IoT vision with traditional Building Automation Systems (BAS). We consider several sensing paradigms and demonstrate the use of NDN to securely interact with NDN-enabled sensors. In the process, we address some challenges caused by sensors' intermittent availability, power constraints and asynchronous communication patterns. Our results include concrete protocols that facilitate secure sensor-bound communication over NDN.

I. Introduction

Today's Internet was designed as a means of communication between two remote hosts. This was envisioned in 1970-s and early 1980-s as the predominant paradigm for resource sharing. Today, common Internet usage scenarios have changed significantly and include: large-scale content distribution, delay-tolerant networking, swarms of wireless devices producing and consuming data as well as mobile computing. Although all these "new" scenarios are addressed to some extent at different layers in the current Internet (e.g., via CDNs, 802.x, DHCP, mDNS), there are several large research efforts that aim to construct and demonstrate new Internet architectures that would better address the needs of modern Internet applications.

Named-Data Networking (NDN) [30] is one such effort. NDN is part of a field typically called Information-Centric Networking (ICN) [18], [26] – consisting of networking paradigms designed for efficient data distribution [22]. Related prominent efforts are PARC's Content-Centric Networking (CCN), whose open source software CCNx is used as a reference implementation for the work described below. In NDN, named data – rather than a named host or an interface address – is a first-class entity. Data is directly addressable, regardless on what host distributes it, and is returned in response to explicit requests from *consumers*, and is never sent unsolicited, i.e., NDN is a "pull" architecture.

NDN also stipulates that each piece of data must be signed by its producer. This allows decoupling of trust in data from trust in the entity that stores and/or disseminates that data. This NDN feature fosters automatic caching of data in order to optimize bandwidth use and enable effective simultaneous utilization of multiple network interfaces.

NDN's long-term goal is to replace the current TCP/IP-based Internet architecture [12]. In order to succeed, it must be shown that NDN can support all major types of communication performed today and envisaged for the near future. Recent work focused on implementing telephony [21], video conferencing [40], smart meters [24], and control systems [4], [34] over NDN.

This paper provides further evidence of NDN's suitability for communication other than data distribution. Specifically, we explore the use of NDN for *secure* sensing applications.

Sensors play a central role in the *Internet of Things* (IoT) [8]. *Smart objects*, which represent the building blocks of IoT, provide a bridge between physical (analog) and digital (cyber) worlds, through sensing. The use of sensors in IoT research builds on research of the embedded and wireless sensing community [2] and more recent efforts in sensing using general purpose mobile devices [5].

Related research in sensing and control by the NDN team, which motivates the work described here, has focused on the context of building automation systems (BAS). BAS are a traditional application of industrial control systems to managing the various systems of buildings, including heating, ventilation and air conditioning (HVAC), electrical distribution, water monitoring, fire detection and suppression, intrusion detection, and access control. The IP protocol suite is increasingly used to network their components and as such is now a fundamental substrate of new buildings [17]. However, IP networks suffer from limitations that impact innovation and trust in networked building systems, which we believe can be addressed with NDN.

In both BAS and IoT scenarios, the main purpose of a typical network-enabled sensor is to collect data and allow other devices and applications to access it remotely. Such sensors tend to be part of resource-constrained devices, for example being either battery-operated or energy-limited for sustainability reasons, with computing resources sufficient to perform data gathering and reporting. In order to save power, sensors can choose to sleep or hibernate whenever possible. As such, our framework supports secure sensor data access in energy-constrained environments.

In such cases where network communication should be

kept to a minimum, the pull nature of NDN communication makes careful protocol design important. Applications must be aware of the availability of new data in order to request it. This is not a problem when data in generated at regular intervals (e.g., daily at 8am). However, applications can not predict the timing of arbitrary external events that trigger sensors to collect new data. For example, a moisture sensor might be triggered by rain or a temperature sensor – by frost.

Sensors are often used in critical environments, e.g., monitoring buildings that are part of critical infrastructure. They can also be used to collect highly sensitive information, e.g., utility smart meters and physical intrusion detection data. Therefore, any general approach to secure sensing must offer availability, integrity, origin authentication and access control (data privacy). Also, due to sensors' limited resources, a common DoS attack vector is to attempt to overwhel the target sensor(s) with malicious requests. Thus, DoS mitigation is an important requirement All of the above, coupled with scalability, represent a major challenge in the context of any Internet architecture, including NDN.

BAS and other industrial control systems have in the past typically employed physical or logical isolation of the network as a primary security measure, and can likely do so no longer [25]. Isolation limits interoperability and integration, and runs counter to both the IoT vision and the reality of how users desire to access data, interconnect heterogeneous components, deliver system upgrades, and monitor performance. Our approach to sensing security must necessarily be more sophisticated.

Focus. This paper focuses on constructing a secure, efficient and scalable sensing framework over NDN. Our framework provides data integrity, flexible access control, efficient multicast communication and basic protection against denial-of-service (DoS) attacks. We classify sensors along two dimensions: data collection and data dissemination, i.e., based on how they obtain and report data.

Data dissemination:

- **Pull:** a sensor does not send out collected data. It is always on-line, ready to accept queries from remote entities.
- **Push:** a sensor automatically disseminates collected data by *pushing* it to a set of remote entities (subscribers).

Data collection:

- **Triggered:** a sensor collects data when triggered by some external event, e.g., whenever smoke is detected.
- **Scheduled:** a sensor collects data at based on some fixed (though not necessarily regular) schedule, e.g., a new temperature reading every minute.
- **On-demand:** a sensor collects data following an explicit request from an external entity.

We then focus on the design of a framework that allows sensors to communicate via *push* and *pull* data dissemination, regardless of how they perform data collection. We believe that this is general enough to subsume a wide range of sensor application settings. Also, our classification is not strict, meaning that a sensor can operate in a hybrid fashion, e.g., a sensor can support both triggered and scheduled collection and implement both push and pull data dissemination modes. The main reason for this classification is to identify main types of sensor communication patterns, which is the first step towards adapting NDN to the world of sensing.

Organization. The rest of this paper is structured as follows: We proceed with an overview of related work in Section II. Then, Section III introduces our security framework, which provides foundation for the sensing protocols, presented in Section IV. The paper concludes in Section V.

II. Background and Related Work

A. NDN Overview

NDN is an example of ICN. Instead of traditional host-centric networking, exemplified by IP (where the exchange of data is a side-effect of conversation between communicating hosts), data is a first-class object in NDN. Each data packet is addressable by a unique name, which can be human readable and globally routable, and is decoupled from the location of its producer. Therefore, any piece of data can be served from any network entity. Mandatory producer-generated digital signatures on all data provide integrity and origin authentication.

NDN defines two types of packets: *interest* and *data*. The latter contains (among other fields) a name, a payload, and a public-key signature. Names are hierarchical, consisting of one or more arbitrary-length components. A name is usually represented as a sequence of components delimited using "/", e.g., `/youtube/video/id/okqEVeNqBhc`. A signature is computed over the name, data, and accompanying signature meta-data associating the signer's public key. Applications are responsible for managing trust bindings between public keys and name-spaces.

Interest packets are used to fetch data by name. A data packet with name X satisfies an interest with name X' if $X = X'$ or X' is a proper prefix of X (i.e., $X = X'/Y$ for some non-empty string Y).

Communication in NDN is receiver-driven. All requested data must be preceded by a corresponding interest. A *consumer* issues an interest for desired data. As the interest is forwarded by each router, it is either satisfied (and consumed) by the router's cache, or propagated further upstream towards the nearest place that stores that data. Data originates at a *producer* which is responsible for publishing it under a *namespace*, injecting it into the network upon receipt of an interest. Data traverses the reverse path taken by the preceding interest.

Packet Forwarding. An NDN router forwards interests and data packets, and caches forwarded data. Each router is expected to implement the following data structures: Forwarding Information Base (FIB), Pending Interest Table (PIT), and Content Store (CS).

FIB is the analog of IP's forwarding table. It is populated with $(name\_prefix, interface)$ entries, which correspond to reachable paths for data produced under the namespace *name_prefix*. Interest lookups are performed using longest-prefix match. Multiple entries for the same prefix are permitted, allowing data to be retrieved from multiple distinct paths.

PIT is used to store information necessary to forward data back to consumers. A PIT entry contains an interest and one or more arrival interfaces for that interest. When an interest is received by a router for the first time, a PIT entry is created. Subsequent closely spaced interests with the same name are aggregated, and the corresponding PIT entry is updated to account for the additional arrival interfaces. Identical interests are not propagated further. Data that satisfies an interest consumes the corresponding PIT entry and is sent out to all associated arrival interfaces.

CS is used to cache data. This offers several benefits: efficient dissemination of data packets, and implicit multicasting for popular data, increasing bandwidth efficiency and reducing latency for downstream consumers. Data packets carry a *freshness* field, which tells routers when particular packets should be flushed from CS.

Authenticated Interests. In contrast with data packets, NDN interests are not authenticated. This is done for privacy and performance reasons. However, NDN imposes virtually no restrictions on the composition of name components. Previous work [4], [10] took advantage of this freedom by embedding binary data as the last name component of an interest. This allows a consumer to efficiently "push" authenticated data towards a producer.

Access Control. In NDN, access control is enforced via encryption of data. Though NDN transparently supports encrypted data, it does not specify any particular access control scheme. CCNx [7], a reference of NDN, provides a basic form of access control, which mimics the file-system access control model [35]. This is a poor match for sensing applications.

B. Sensing Overview

Our consideration of sensors draws from the Internet of Things (IoT) research and embedded and wireless sensing before it, as well as the field of Building Automation Systems (BAS). We consider NDN-based building networks as a continuation of the convergence of traditional building automation with IP networking and other enablers of the IoT vision. Building Automation Systems (BAS) have more concrete security requirements than IoT in general, especially when they are related to critical infrastructure.

Sensor data is usually communicated to other sensors, applications or data collection nodes – known as a sinks. In what follows, we consider sensors that for reasons of energy conservation and/or cost have very low computing power and significant energy limitations. Research in wireless sensor networks (WSN) is broad and extensive, covering application case studies [2], [29], [39], energy efficient link-layer protocols [2], [23], [27], [39], routing protocols [1], [3], key establishment [6], [28], [11], etc. Relevant to our work, we present an overview of network layer transport for networks similar to WSN assuming homogeneous device and discuss security measures used for WSN.

The IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) architecture is an attempt to bring IPv6 to low-powered devices such as sensor networks, specifically tuned for devices implementing IEEE 802.15.4 as the basis of numerous low-powered link-layer wireless protocols such as ZigBee, MiWi, etc. Security of 6LoWPAN is relegated to the link-layer protocols. While IEEE 802.15.4 provides link-layer authenticated encryption (such as AES-CTR), several shortcomings have been pointed out in a comprehensive security analysis in [33]. While at the MAC-layer access control is supported, it is statically assigned by the application and limited to 255 entries.

Estrin et al. [20] introduced Directed Diffusion (DD), the first data-centric approach to sensor networks. DD bears a strong resemblance to NDN in that DD sensor nodes produce location-independent data addressed using names (in this case attribute-value pairs). DD reinforces desired paths using interest-data flow balance feedback, however provides limited resilience towards availability attacks, specifically when the adversary drops interest packets. This however does not assist in interest-flooding attacks or false data injection. The authors of [37] propose a secure diffusion mechanism. It ensures connectivity to authentic sensors while limiting malicious data injection within a local neighborhood. It is based on TESLA [31] broadcast authentication on interests. Nodes endorse sensor data using MAC. The scheme dovetails DD's path reinforcement to ensure paths are selected with known good data, limiting the propagation of malicious/unauthentic data.

Heinzelman et al. [19] propose LEACH (Low-Energy Adaptive Clustering Hierarchy), a clustering-based routing protocol for WSNs that evenly distribute load among all sensors in a network. LEACH provides scalability and robustness via localized coordination. Xiao-yun et al. [38] introduce a secure extension to LEACH, called SLEACH, which uses lightweight cryptography, such as hash chains, to provide authentication of LEACH messages. However, the paper does not address access control or data integrity.

In [32] Perrig et al. introduce SPINS, a suite of building blocks for WSNs. Spins provide facilities for authentication (broadcast and one-to-one), confidentiality and freshness. The broadcast authentication protocol is based on TESLA [31]. Similarly to our work, confidentiality is obtained through encryption. However, SPINS does not provide a framework for flexible access control.

III. Secure Sensing Framework

In this paper, we take advantage of prior work that proposed a security framework for lighting control over NDN [4]. Lighting control is specific case of building automation which, in turn, is a representative example of actuation and control. Since sensing and actuation are closely related, they naturally share most security features – though there are some key differences as well.

One obvious distinction is that a sensor, unlike an actuator, does not usually affect the physical environment. Another difference is that, in general, an actuator accepts a command, executes it (e.g., by changing state) and returns its status. Whereas, in either push or pull dissemination model, a sensor reports (possibly large amounts of) data collected from the environment. Finally, an actuator is generally controlled by one or a few entities, e.g., a typical light fixture is controlled by one or two switches. In contrast, a multitude of entities

might need to retrieve data from a single sensor. This imposes a very different scalability requirement.

Our setting includes the following parties: a configuration manager (CM), a set of sensors (Sen), multiple applications (App), and an authorization manager (AM). CM is responsible for initial configuration of Sen, which involves: (1) assigning Sen an NDN namespace ($name_{\mathsf{Sen}}$), under which Sen will publish its readings; (2) assigning an identity to Sen represented by a unique public/private key pair ($pk_{\mathsf{Sen}}, sk_{\mathsf{Sen}}$); and (3) installing AM's public key (pk_{root}) on Sen. pk_{root} identifies the root of trust shared by all sensors and applications within the same domain.

Framework Components. Our sensing framework provides a trust model, a syntax for finely granular access control policies, and a set of protocols. The trust model associates keys (and their owners) to namespaces, allowing parties to determine whether data published under a namespace is signed by the correct entity.

Similar to sensors, applications are identified by their public/private key-pairs. When an application joins a domain (i.e., it is allowed to query a subset of the sensors available in that domain), AM signs the public keys for App and optionally assigns it a namespace ($name_{\mathsf{App}}$). For each sensor Sen, AM distributes a signed access control list containing the identities and namespace of applications which are allowed to query Sen. While CM and AM represent separate functionalities, in practice can correspond to the same physical entity. Without loss of generality, we assume that all data produced by sensors is confidential. We therefore enforce access control on it.

To formalize our security goals, we consider an adversary with full control over the communication channel between App and Sen. We believe this is a realistic scenario, since sensor data is often collected via wireless networks or otherwise accessible channels. Additionally, the NDN architectural vision promotes data-centric security over channel-centric security [36], so assuming an insecure channel is appropriate even in wireline networks. The goals of the adversary include: (1) forcing Sen to respond to unauthorized queries (including old legitimate queries from App); (2) sending incorrect data to App in response to a legitimate query; (3) sending illegitimate acknowledgements (when available) to Sen, claiming receipt of data; and (4) violating data privacy by gaining unauthorized access to Sen's readings.

A. Trust Model

The trust model ensures that entities only publish data to their assigned namespace, or children namespaces. ($name_A$ is a child of $name_B$ then the latter is a prefix of the former.) This is done by associating one or more public keys with each namespace. A data packet published under $name_A$ must be signed using the key associated to $name_A$ or any of its ancestors.

AM, acting as a trusted-third party (TTP), generates a public/private key-pair $K_{root} = (pk_{root}, sk_{root})$ and distributes pk_{root}. This key serves as the root of trust: valid keys must be either signed using sk_{root}, or another valid key. To associate producer P (e.g., an application or a sensor) with a namespace $name_P$, AM publishes public key pk_P, under $name_P/\texttt{key}$. This can be seen as a simple public-key certificate. P can further delegate its namespace, assigning a child namespace $name_{P'} = name_P/\texttt{sub-namespace/}$ to P' by publishing (and therefore signing) P''s public key under $name_{P'}/\texttt{key}$.

Proving ownership of $name_P$ is trivial using a challenge-response protocol. The challenger issues an interest for $name_P/\texttt{nonce}$ where `nonce` is a randomized string selected by the challenger. P can respond with a valid data packet only if it owns the signing key linked to $name_P$, any of its ancestors, or AM's signing key. Furthermore, in this trust model, the set of keys that can sign data belonging to namespace $name_P$ (or children namespaces) can be unambiguously determined, i.e., contains all valid keys with prefix $name_P$ and the root key.

B. Key Attributes and Access Control Policies

The attributes of a public key are expressed as the name under which the key is published. Attributes are name/value pairs expressed as two consecutive name components: the first indicates the attribute name followed by a component containing its value. Following our trust model, the key is signed by either the AM or the owner of the namespace.

The following lists a sample set of attributes which can be used:

- `domain`: the application domain
- `appname`: the application identifier
- `access`: application permission to query a sensor
- `expires`: expiration date specified in *generalized time* notation (YYYYMMDDHHMMSSZ)

For example, public key pk_P can be published as:

```
/uci/ics/domain/smokealarm/appname/
 campuswarning/access/full-access/
    expires/20201231235959Z/key
```

C. Bootstrap and Pairing Protocol

New sensors must be *paired* with CM and bootstrapped before they can accept commands from applications. Pairing involves installing pk_{root} on Sen. During bootstrap, CM specifies a namespace for Sen and sets Sen's clock. The public key identifies the domain in which Sen operates. CM proceeds to send the signing key to Sen. This key-pair links Sen to its assigned namespace. CM may additionally communicate the names for one or more ACLs which Sen can use to determine applications' permissions. Sen generates a symmetric key k_{Sen}, which can be used to further derive application-specific symmetric keys. These keys are used for authentication purposes. After Sen is setup, it responds to CM with the current time and a hash of all information exchanged during the bootstrap.

D. Using Symmetric Cryptography for Higher Efficiency

So far we have only considered public-key cryptography for encryption and authentication. Since we assume that might have very limited computing power, public-key cryptography may be too expensive. To reduce the cost of encryption and

authentication, we briefly illustrate how to use symmetric cryptography between Sen and App.

As a first approach, Sen and App could simply share a symmetric key. This key is used (directly or after being transformed by a key derivation function) for authenticating and encrypting messages between the two parties. To allow simple revocation and meaningful authentication, Sen should have a separate key for each App. However, if a large number of applications interact with Sen, then the amount of key material that Sen needs to store may be prohibitively large – linear in the number of Apps. Instead of storing all keys, Sen can compute them when needed using a pseudo-random function (PRF), keyed with a secret key known only to Sen. Key k_{App} – the symmetric key corresponding to App – is computed as $k_{\mathsf{App}} = \mathrm{PRF}_{k_{\mathsf{Sen}}}(name_{\mathsf{App}})$. This key must be sent to App during its first interaction – possibly secured using public-key cryptography – with Sen. The key can then be used to derive encryption and authentication keys.

IV. Secure Sensing Modalities

In this section we introduce the two modalities for data dissemination: *pull* and *push*. Each modality represents a class of sensors with different scope and capabilities. By addressing them separately, we aim to design a set of protocols suitable for a wide range of applications.

A. Pull Data Dissemination

Sensors implementing pull-based data dissemination are assumed to be always on-line, and therefore can efficiently employ the standard protocol for data dissemination in NDN:

1) App expresses an interest int_{Sen} for $name_{\mathsf{Sen}}$.
2) After receiving int_{Sen}, Sen performs the following actions: (1) (possibly) senses the environment and generates data; and (2) returns a (signed) data packet C_{Sen} containing the data.
3) App receives C_{Sen}, verifies the signature, and uses the data. Since the signature in C_{Sen} binds name with data, App can verify that the data returned corresponds to its most recent request.

Access control is enforced by Sen using data encryption. In particular, data packets are encrypted by Sen using App's public key, which is certified (i.e., signed) by AM. (As an alternative, Sen can encrypt data using App's symmetric key, as discussed in Section III-D.) Upon receipt of a data packet, App decrypts it using its secret key. For efficiency reasons, data is encrypted using *hybrid* encryption [14]: the data is first encrypted using a symmetric encryption scheme (e.g., AES in CBC mode) under a random key k; k is then encrypted under the authorized recipient's (public or symmetric) key. In case multiple applications have access to the same sensor reading, k is encrypted under all recipients' keys:

$\mathrm{E}_{pk_1}(k)$	$\mathrm{E}_{pk_2}(k)$	$\mathrm{E}_{pk_3}(k)$	$\mathrm{E}_k(m)$

This allows the producer to encrypt the data only once, and requires it to simply encrypt a (relatively small) key once per recipient. The cost of this approach, both in terms of computation and communication overhead, is reasonable when the number of recipients is small (e.g., up to 20-30). If a large number of recipients need access to the data, efficient broadcast encryption (BE) [13], [15] can provide constant (instead of linear) ciphertext expansion and encryption time.

Flexible Query Authentication. In dynamic environments, multiple applications may be frequently added to the list of parties authorized to query a particular set of sensors. In this case, updating the ACL and corresponding application public keys on all affected sensors may be cumbersome and expensive. As an alternative, we introduce a flexible query authentication mechanism that does not require any direct communication between AM and the affected sensors.

Application App that owns a key distributed under name $name_{\mathsf{App}}/key$, where $name_{\mathsf{App}}$ is the namespace owned by App, issues an interest to Sen as: "$name_{\mathsf{Sen}}/name_{\mathsf{App}}/$`auth-token`".

The field `auth-token` encodes an authentication token, constructed as: `state || authenticator`. The `state` represents information required to prevent timing and replay attacks. It is composed of a sequence number, timestamp, and estimated round-trip time (RTT) between App and Sen. The `authenticator` part is a signature or a MAC, computed over $name_{\mathsf{Sen}}/name_{\mathsf{App}}/$`state`. App authenticates the query using either an established symmetric key (to compute the MAC) or public key associated with $name_{\mathsf{App}}/$`key` (to generate the public key signature).

When Sen receives the interest, it performs the following:

1) Sen verifies the state corresponding to App, determining if the interest is current. If it has no record of previous queries from App, it extracts the sequence number from `auth- token` and stores it as tuple $(name_{\mathsf{App}}, sequence\_number)$. Otherwise it checks the stored sequence number for App is lower than the one in `auth-token`.
2) Sen verifies the signature or MAC contained in `authenticator`. If it is a signature and Sen has not yet retrieved the public key, it retrieves it from $name_{\mathsf{App}}/key$ and stores it in a local cache.
3) Sen senses the environment, generates data, and encrypts it under the public key pk_{App} or a symmetric encryption key derived using k_{App}. The data is encapsulated in a data packet under the same name as the interest, and signed under pk_{Sen} or MAC-ed using the output of a key derivation function over k_{App}.

If the pair $(name_{\mathsf{App}}, sequence\_number)$ stored by Sen is not updated for a pre-determined amount of time, it is considered stale and deleted. This ensures Sen only retains state for currently active applications.

Mitigating Denial-of-Service Attacks. The protocols introduced in this section are designed for anemic sensors, which can be accessed over the network by both trusted and untrusted applications. A sufficient number of requests from a malicious party could easily prevent legitimate applications from querying the sensor and/or quickly deplete the sensor's battery.

To prevent Sen from being overwhelmed with queries, the cache of intervening routers can be used to handle closely

spaced interests, reducing the bandwidth and delay for responding to App. Sen can specify the optional `Freshness` field in produced data packets to control the expiry timer of data in cache. As an example, this value can be adjusted according to the amount of time necessary to process queries. Subsequent closely spaced interests from App will be satisfied by the cache until Sen is able to issue new readings. On the other hand, if Sen must perform separate sensing for each query, the `Freshness` field in data packets can be set to 0, so that intermediate routers immediately mark the data packets as stale, and remove them from their cache.

This approach works only if applications request data from the sensor using the same data name: as an example, if each application includes a unique identifier within the interest name, then subsequent requests carrying different identifiers cannot be satisfied using the cache. Moreover, malicious requests can be crafted in such a way that they can never be satisfied using routers' caches: as an example, the adversary can append a random nonce at the end of the interest name.

As an alternatives, sensors may accept only interests authenticated via MAC. This way, a sensor can determine if a query is legitimate by performing a key derivation from an application-provided string of bounded length, and single operation based on (efficient) symmetric cryptography. Non-legitimate interests will be immediately discarded, without requiring any sensing, encryption or signature computation over sensed data. While DoS attacks are not warded off in their entirety by using authenticated interests, this ensures that work is expended only if the source is permitted.

B. Push Data Dissemination

With push, Sen senses and generates data when triggered by an asynchronous event. Asynchronous events are tightly correlated with failure or important notification, and therefore Sen must attempt to reliably send data to App. However App retrieving data from Sen proves problematic. Recall that NDN adheres to the "pull" paradigm, i.e., it mandates that all communication is receiver-driven. Therefore App, as the recipient of the data, must initiate communication to Sen in order to receive its data.

One option is to use the above pull protocol, and have App poll Sen by repeatedly issuing interests to $name_{\mathsf{Sen}}$ at repeated time intervals. When an event occurs, Sen awakens, senses and returns data encapsulated in a data packet satisfying any outstanding interests. However, between each triggered event, resources are tied up in intervening routers to store outstanding interests issued by App. App must also expend significant resources in refreshing interests. Furthermore, App may incur large delay for receiving data from Sen if it must poll many devices. Delay can occur when Sen's event occurs between poll intervals while App is continuing to poll other devices. The resultant delay is therefore proportional to the number of devices on the network. For environments which must ensure timely delivery after event occurrence, this may not be a suitable approach. Ideally, Sen would notify App when new data is available, prompting App to query Sen. However, NDN does not provide explicit support notifications. Nevertheless, interests can be used to implement such a mechanism.

Interest Notification. App listens to a distinct namespace $name_{\mathsf{App}}$ which is assigned by AM. The signed access control list includes each namespace $name_{\mathsf{App}}$ for the corresponding identity pk_{App}. Each application listens for interests on their corresponding namespace. Sen must have advanced knowledge of which namespace has been assigned to each application, and notifies each application separately. Sen reaps the benefits of NDN implicit broadcast ability by generating and transmitting content C_{Sen} once. We now describe the protocol in detail:

1) Sen (1) senses the environment; (2) generates data; (3) enforces access control via data encryption; and (4) generates a signed data packet C_{Sen}, embedding the data with the name $cname$ under the namespace $name_{\mathsf{Sen}}$.
2) For each application: Sen issues interest int_{App} for $name_{\mathsf{App}}/[cname]$, where $[cname]$ is an opaque name component containing the full name of C_{Sen}.
3) App receives int_{App}, parses $cname$, and sends interest int_{Sen} for $cname$.
4) Sen receives int_{Sen}, and responds with C_{Sen}.
5) App receives C_{Sen}, verifies the signature, and decrypts the data using its secret key.

Sen can encrypt and embed data in an authenticated interest when notifying App. This way, applications do not need to wait a full RTT from the time they receive the notification to the time they acquire the sensor reading. Instead Sen only needs to issue an interest with the name: "$name_{\mathsf{App}}$/`data`/`auth-token`". The field `auth-token` is encoded as `timestamp || authenticator`. The `timestamp` is used to prevent replay attacks. The `authenticator` part is a signature or a MAC computed over $name_{\mathsf{App}}$/`data`/`timestamp`. The `data` is the new data ready to be sent to App, and is encrypted under pk_{App} or k_{App} using name encryption, that is, encrypting the data and embedding it in one of the rightmost name components.

On receipt of the authenticated interest, App ensures the timestamp is current and proceeds to verify the signature and decrypts data. Additionally, App can sign empty data in response to notifications as an acknowledgement that the data is received. If acknowledgements are required, Sen must retain state for each acknowledgement not yet received from App.

For a low-powered sensor, ensuring delivery of data is an intensive task specifically due to the energy and bandwidth. Each notification using the protocol as described above requires potentially expensive cryptographic operation for both encryption and signing. This cost is proportional to the number of applications and requires Sen to store a static list to notify App. By introducing a *proxy node*, which facilitates communication between Sen and App, Sen can limit connectivity to a single proxy using the same protocols as it would with App. A proxy is equipped with a fixed power source and more computing resources. The proxy can also retain a backlog of data received by Sen for later querying by App using the pull protocol.

Resilience against DoS Attacks. In the push data dissemination protocol, sensing and corresponding data delivery is not triggered by an interest. Therefore, Sen cannot be victim of

DoS attack using the flooding mechanisms illustrated above. On the other hand, with interest notification, App can be inundated with interests, but this requires an adversary to issue interests to each target namespace. The techniques in [16], [9] can be used to detect and mitigate such interest flooding attacks.

V. Conclusion

With the advent of the IoT and its convergence with critical infrastructure such as building automation systems (BAS), secure connectivity of resources constrained devices is becoming increasingly important. This paper focused on the design of a secure and efficient framework for connecting sensors with applications over NDN. Our framework includes a trust model that allows parties to authenticate sensor data, and fine-grained access control mechanisms based on data encryption as well as key attributes. We considered three types of sensors and constructed corresponding communication protocols tailored for NDN.

References

[1] K. Akkaya and M. Younis. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3, 2005.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38, 2002.

[3] J. Al-karaki and A. Kamal. Routing techniques in wireless sensor networks: A survey. *IEEE Wireless Communications*, 11, 2004.

[4] J. Burke, P. Gasti, N. Nathan, and G. Tsudik. Securing instrumented environments over content-centric networking: the case of lighting control and NDN. In *INFOCOM NOMEN*, 2013.

[5] J. A. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory sensing. In *World Sensor Web Workshop, ACM Sensys 2006*, 2006.

[6] S. Camtepe and B. Yener. Key distribution mechanisms for wireless sensor networks: a survey. Technical report, 2005.

[7] Content centric networking (CCNx) project. `http://www.ccnx.org`.

[8] M. Chui, M. Löffler, and R. Roberts. The internet of things. *McKinsey Quarterly*, 2:1–9, 2010.

[9] A. Compagno, M. Conti, P. Gasti, and G. Tsudik. Poseidon: mitigating interest flooding ddos attacks in named data networking. In *LCN*, 2013.

[10] S. DiBenedetto, P. Gasti, G. Tsudik, and E. Uzun. ANDaNA: Anonymous named data networking application. In *NDSS*, 2012.

[11] L. Eschenauer and V. Gligor. A key-management scheme for distributed sensor networks. In *CCS*. ACM, 2002.

[12] National science foundation (NSF) future of internet architecture (FIA) program. `http://www.nets-fia.net/`.

[13] A. Fiat and M. Naor. Broadcast encryption. In *CRYPTO*, 1993.

[14] E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In *CRYPTO*, 1999.

[15] P. Gasti and A. Merlo. On re-use of randomness in broadcast encryption. In *PST*, 2011.

[16] P. Gasti, G. Tsudik, E. Uzun, and L. Zhang. DoS & DDoS in Named-Data Networking. In *ICCCN NACSD*, 2012.

[17] W. Granzer, D. Lechner, F. Praus, and W. Kastner. Securing ip backbones in building automation networks. In *Industrial Informatics, 2009. INDIN 2009. 7th IEEE International Conference on*, pages 410–415. IEEE, 2009.

[18] M. Gritter and D. Cheriton. An architecture for content routing support in the internet. In *USENIX USITS*, 2001.

[19] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. Energy-efficient communication protocol for wireless microsensor networks. In *HICSS*, 2000.

[20] C. Intanagonwiwat, R. Govindan, D. Estrin, J. S. Heidemann, and F. Silva. Directed diffusion for wireless sensor networking. *IEEE/ACM Trans. Netw.*, 11(1):2–16, 2003.

[21] V. Jacobson, D. Smetters, N. Briggs, M. Plass, J. Thornton, and R. Braynard. VoCCN: Voice-over content centric networks. In *ReArch*, 2009.

[22] V. Jacobson, D. Smetters, J. Thornton, M. Plass, N. Briggs, and R. Braynard. Networking named content. In *ACM CoNEXT*, 2009.

[23] C. Jones and K. Sivalingam. A survey of energy efficient network protocols for wireless networks. *Wireless Networks*, 7, 2001.

[24] K. Katsaros, W. Chai, N. Wang, G. Pavlou, H. Bontius, and M. Paolone. Information-centric networking for machine-to-machine data delivery: a case study in smart grid applications. *Network, IEEE*, 28(3), May 2014.

[25] E. D. Knapp. *Industrial Network Security: Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems*. Syngress, 2011.

[26] T. Koponen, M. Chawla, B. Chun, A. Ermolinskiy, K. Kim, S. Shenker, and I. Stoica. A data-oriented (and beyond) network architecture. In *ACM SIGCOMM*, volume 37, pages 181–192. ACM, 2007.

[27] S. Kumar, V. Raghavan, and J. Deng. Medium access control protocols for ad hoc wireless networks: a survey. *Ad Hoc Networks*, 4:326–358, 2006.

[28] D. Liu and P. Ning. Establishing pairwise keys in distributed sensor networks. In *CCS*. ACM, 2003.

[29] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson. Wireless sensor networks for habitat monitoring. In *ACM WSNA*, 2002.

[30] Named data networking project (NDN). `http://named-data.org`. Retrieved Mar. 2013.

[31] A. Perrig, R. Canetti, J. D. Tygar, and D. X. Song. Efficient authentication and signing of multicast streams over lossy channels. In *IEEE Symposium on Security and Privacy*, pages 56–73, 2000.

[32] A. Perrig, R. Szewczyk, J. D. Tygar, V. Wen, and D. E. Culler. Spins: Security protocols for sensor networks. *Wireless Networks*, 8(5):521–534, 2002.

[33] N. Sastry and D. Wagner. Security considerations for IEEE 802.15.4 networks. In *WiSec*. ACM, 2004.

[34] W. Shang, Q. Ding, A. Marianantoni, J. Burke, and L. Zhang. Securing building management systems using named data networking. *Network, IEEE*, 28(3), May 2014.

[35] D. Smetters. CCNx access control specifications. Technical report, PARC, 2010.

[36] D. Smetters and V. Jacobson. Securing network content. *Technical Report TR-2009-1, Xerox Palo Alto Research Center-PARC*, 2009.

[37] X. Wang, L. Yang, and K. Chen. SDD: secure directed diffusion protocol for sensor networks. In *Security in Ad-hoc and Sensor Networks*, volume 3313 of *LNCS*, pages 205–214. Springer, 2005.

[38] W. Xiao-yun, Y. Li-zhen, and C. Ke-fei. SLEACH: secure low-energy adaptive clustering hierarchy protocol for wireless sensor networks. *Wuhan University Jurnal of Natural Sciences*, 10(1):127–131, 2005.

[39] J. Yick, B. Mukherjee, and D. Ghosal. Wireless sensor network survey. *Computer Networks*, Aug. 2008.

[40] Z. Zhu, J. Burke, L. Zhang, P. Gasti, Y. Lu, and V. Jacobson. A new approach to securing audio conference tools. In *AINTEC*, 2011.