Securing IoT Networking with ICN

Dhruti Lalaji Sawant (Matrikel Nummer: 5156829)

***Abstract*—Internet of Things (IoT) is connecting our physical world with the Internet e.g. connected automobiles, e-Health, smart homes , industrial automation et cetera. Information-centric Networking (ICN) is a concept that changes the Internet design which focuses on transmitting and accessing named content instead of the traditional host-based IP data. Several challenges present itself when we consider the dynamic infrastructure and deployment of IoT environment such as auto-configuration of the names,enabling push traffic and minimizing header size. However, the most significant challenge is to ensure the security and privacy of sensitive data transferred over the network. This report discusses the ICN-IoT middle-ware which provides a design level solution to the challenge of security and privacy in communication of data.**

I. Introduction

With the current developments in technology, Internet of Things(IoT) seems to have entered into every sphere of everyday life. From our homes to our cars to hospitals and other public and private institutions. Small heterogeneous embedded devices that are connected to each other wirelessly to provide services that were unimaginable before. However, the most important aspect of any service is to maintain security of the processed data and to maintain the privacy of the users. Hence the IoT services tend to implement dedicated security measures.

However, the Internet itself is evolving and now we see the advent of a new approach to the internet paradigm known as Information-centric networking (ICN). The new Internet concept revolves around the idea of not addressing the content on a particular device through IP addresses but through assigned name. The problem lies in the fact that IoT devices are very well expendable and to assign the myriad number of these devices with limited IP addresses and to create mappings on several routers is computationally expensive. These IP addresses have nothing to do with what content the device holds. It is only a unique identifier that is used by the network layer to communicate with other devices.

So what does this name-based addressing mean? The ICN infrastructure introduces the concept of naming the devices based on the content it holds or through the service it can access. This allows the data stored in any of these devices to become completely independent of the devices location, the application it is running on, the storage format, the means of transportation thereby reducing all these overheads of the data packet. The ICN favors the concept of caching and replication of data close to the service or the device where it is required. This allows for higher efficiency, better scalability and higher robustness of data in challenging communication scenarios.

This report discusses about how this ICN technology could be relevant for an IoT environment. We would see what features of ICN favor the networking in IoT devices. This will be discussed in the II-A, where the ICN protocol stack will be discussed. Then we will discuss about the ICN middle-ware architecture that could potentially provide us with authenticity, confidentiality, integrity and privacy in II-C and thereafter the later sections will discuss with the help of UML diagrams, how these functionalities can be provided to our ICN-IoT system.

First and foremost, ICN offers features such as caching, computing and storage, mobility, context-aware networking and support for ad hoc networking. Therefore it allows IoT devices a platform that supports device bootstrapping with minimal configuration, self organization capabilities of IoT elements and natural caching points in the network for faster retrieval of data.

Another aspect of ICN is that the data in this information-centric networking is delocalized and does not require retrieval via an end-to-end transport stream but instead with hop-wise replication and in-network caching, this facilitates information dissemination in the IoT environment, and relaxes the demand for continued connectivity.

These concepts can however be coupled with the existing Internet. ICN doesn't mean that Internet will be replaced, it only facilitates the features mentioned however it is easily implemented on top of the current IP infrastructure. The key point is that such a concept can be applied to different layers of the protocol stack to provide resource naming, ubiquitous caching and corresponding transport services. This naming functionalities would make fundamental changes in how the current routing and forwarding works.

This being said, it is important to state that the current IP stack has a lot of overheads due to the increasing security measures it must implement. This reduces the overall efficiency and performance of the system. In order to combat that, ICN itself must be implemented with security in mind so that we do not have to rely on these overheads. The host centric trust model depends on retrieving data from a trusted server via a secure connection, this however does not work when we consider heterogeneous IoT devices. ICN has a better solution, it implements security, trust, privacy functions directly to the information content of the device itself. So the most important aspect of the system is not the device but the content itself. The ICN provides with a name to this data and implements signatures to create a trust model. The name is also provided given a set of authorization rules to ensure legitimate accessibility.

As discussed ICN can be implemented at different layer of the protocol stack. When we look at the traditional host-centric systems, the network IP addresses where mapped to applications and services but the mapping was always inconsistent. Therefore when an IoT application wants to access a particular data or services it has to go through the entire mapping of IP addresses to retrieve this information. This

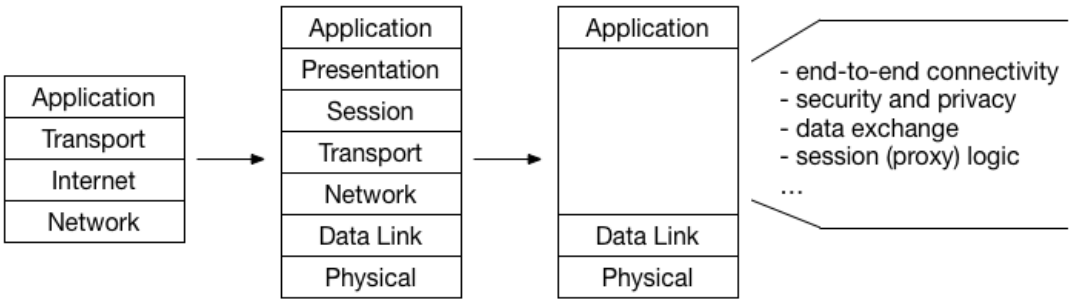

Fig. 1. TCP/IP stack

can be avoided by letting ICN name the data and services in such a manner that the network layer can have an easier or faster access to this information. Communication with a specific device is often secondary, but when needed, the same ICN naming mechanisms can be used. The network distributes content and provides a service, instead of only sending data between two named devices. In this context, data content and services can be provided by several devices, or group of devices, hence naming data and services is often more important than naming the devices. This naming mechanism also enables self-configuration of the IoT system.

The most aspect is however that a Name-Data integrity is maintained such that a given name uniquely corresponds to the data or service of the system. Authenticity is provided by the signature based scheme. Confidentiality can be assigned through a set of keys from the application layer. All of this means that the actual transmission of data does not have to be secured as the same security mechanisms protect the data after generation until it reaches the client, regardless of whether it is in transit over a communication channel or stored in an intermediate cache. In an ICN network, each individual information within a stream of immutable data could potentially be retrieved from a cache in a different location. Having a trust relationship with each of these different caches is not realistic. Through Name-Data Integrity, ICN automatically guarantees data integrity to the requester regardless of the location from where it is delivered.

II. Networking in IoT

A. The Protocol stack in ICN

It is after years of networking research and development that we are able to use the TCP/IP stack as the default standard for all networking protocols in this age of information. The TCP/IP stack is designed based on the standard OSI Model of different layers, each with different functionalities that in turn help the functionalities of the upper layers. This is basically a complex system of network protocols directly influenced by the lower abstractions up till the application layer that help connect to different users or applications throughout the world. However from the early days, security was not incorporated in the design of this stack. Therefore the foundation of this IP stack always had room for improvement in the security aspect. But such an improvement costs additional overhead to the system that causes lower performance and further delay. At the rate at which technology is advancing computing power has become cheaper but with the advent of IoT devices and the scale at which it has encroached our day to day life requires a second look at the way we use this network stack.

In doing so, it was noted that our current applications do not access devices based on who is using it (as it used to in earlier office rooms.) but instead due to globalization, the applications are more content oriented, a user is more likely to find a certain piece of information or content on the Internet rather than from whom it comes. This observation led to the fact that when a user tries to access such an information on the Internet, the network layer goes through several mapped translations before it can send the user this information. This is computationally more work than what it would have been if this data was asked directly. This is where the ICN paradigm comes to play.

The ICNs network design is application driven. The current applications such as facebook, spotify , netflix et cetera provide users with some sort of content over the Internet. This content is bound to some sort of identifier which is accessed through the Internet. But is this really efficient? The answer is no. This static information can be referenced in a simpler manner with names rather than creating identifiers over the networks for routing tables to forward around. Thus ICN allows the network layer to have an abstraction that accesses named content rather than addressable end-hosts. In doing so, it also adds the requires security measures to it.

It is however crucial to have a separate naming abstraction to provide for globally unique and content relevant naming abstraction above the ICN networking layer. These names not only help to find the location of the device faster but also states which services is this device allowed to access according to the access mechanisms.

Here is the traditional stack and its features in Fig. 1. The IP stacks layers have a network layer (Layer 3), upon that there is the transport layer (Layer 4) that provides end-to-end communication and on top of this there is another layer that proves a secure channel for communication. As these layers grew to add more functionality, the applications required more APIs to handle the host-centric Internet. Since most of the communication occurred with a simple HTTP get request, all the layers underneath became depended on each other and became almost static in nature. But the idea was always to fetch a particular data and not to reach a particular

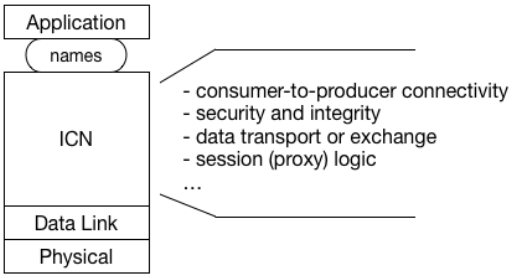

Fig. 2. ICN stack

device. Hence ICN is an ambitious goal to redevelop the layers between the data link and application layer in such a way which is best suited for today's applications that are content driven.

In Fig. 2, we can see how these layers are merged into a single ICN layer that has a name abstraction before the application layer. The functionalities that are provided are thus, consumer to producer connectivity, security with integrity, data transport and a session logic. This eradicates the technical debt of the network layer of the IP stack.

B. IoT Architectural Requirements

First, we will look at what an IoT systems architectural requirements are. They are as follows:

1) Naming: The names assigned to each device must be globally unique. In an IoT environment where the devices are mobile or migrate from one place to another and have a questionable lifetime. It is important that the names remain persistent. Therefore more stress on what the device holds as content rather than its locations. Another aspect of naming is that they must be self certifying. That is the names assigned to a particular device must be bound to that particular device and this must be true at all times. This certification must be made by the device itself. In ICN, users may either generate their own public keys and submit them to the Name certifying service(NCS) for registration, or may contact the NCS to acquire public keys. This is based on a trust model that the NCS will certify the binding of the name and device and thus provide integrity.

2) Security and Privacy : The security challenges are data integrity, authentication of users, access control assigned to devices or devices, privacy of data producers and consumers, privacy of the data content or service responses, privacy of contextual information such as time and location of data transmission. These must be included in the system without interfering with the flexibility and usability of the application by users.

3) Scalability : Cisco predicts there will be around 50 Billion IoT devices such as sensors, RFID tags, and actuators, on the Internet by 2020. As mentioned above, a unified IoT platform needs to name every entity such as data, device, service etc. Scalability has to be addressed at multiple levels of the IoT architecture including naming, security, name resolution, routing and forwarding level. Mobility adds further challenge in terms of scalability. Particularly with respect to name resolution the system should be able to register, update or resolve a name within a short latency.

4) Resource Constraints : End to end latency is affected by the device constraints such as power, storage, computing constraints. Since the IoT devices are usually low end devices, their constraints must be considered before applications can be successfully be run on them. This especially plays a role in real time applications.

5) Traffic Characteristics : Another form of constraint is traffic constraint. This is due to the presence of neighboring devices that require data filtering or due to wide area update where a broad knowledge base is accumulated. As a result, the traffic characteristics of the IoT services have to be properly accounted for during provisioning the system.

6) Contextual Communication : Many IoT applications rely on dynamic contexts in the IoT system to initiate, maintain and terminate communication among IoT devices. Here, we refer to a context as attributes applicable to a group of devices that share some common features, such as their owners may have a certain social relationship or belong to the same administrative group, or the devices may be present in the same location. These temporary groups are referred to as contexts. IoT applications need to support interactions among the members of a context, as well as interactions across contexts.

7) Handling Mobility : There are several degrees of mobility in different IoT scenarios, ranging from static as in fixed assets to highly dynamic in vehicle- to-vehicle environments. Mobility in the IoT architecture can mean either the data producer mobility (i.e., location change), the data consumer mobility, IoT Network mobility (e.g., a body-area network in motion as a person is walking); and disconnection between the data source and destination pair (e.g., due to unreliable wireless links). The requirement on mobility support is to be able to deliver IoT data below an application's acceptable delay constraint in all of the above cases, and if necessary to negotiate different connectivity or security constraints specific to each mobile context.

8) Storage and Caching : Storage and caching plays a very significant role depending on the type of IoT ecosystem, also a function subjected to privacy and security guidelines. Caching is usually done for increasing data availability in the network and reliability purposes, especially in wireless scenarios in the network access and with intermittent connectivity to the infrastructure network. Storage is more important for IoT, storing data for long term analysis.

Data is stored in strategic locations in the network to reduce control and computation overhead. Furthermore considering hierarchical nature of IoT systems, ICN architectures enable flexible heterogeneous and potentially fault-tolerant approach to storage and caching providing contextual persistence at multiple levels.

9) Communication Reliability : Reliable communication desires the following capabilities for the underlying system: seamless mobility support under normal operating conditions, efficient routing in the presence of intermittent disconnection, QoS aware routing, support for redundancy at all levels of a system (device, service, network, storage etc.), and support for rich and diverse communication patterns, both within an

IoT domain consisting of multiple IoT nodes and one or more gateway nodes to the Internet and across multiple such domains.

10) Self-Organization : Self organization is the capability to quickly discover heterogeneous and relevant (local or global) devices/data/services based on the context. This discovery can be achieved through an efficient publish-subscribe service, or through private community grouping/clustering based upon trust and other security requirements.

11) Ad hoc and Infrastructure Mode : The unified IoT architecture needs to design a common protocol that serves both modes. Such a protocol should address the challenges that arise in these two modes: (1) scalability and low latency for the infrastructure mode and (2) efficient neighbor discovery and ad-hoc communication for the ad-hoc mode.

12) IoT Platform Management : An IoT platforms' service, control and data plane will be governed by its own management infrastructure which includes distributed and centralized middle-ware, discovery, naming, self-configuring, analytic functions, and information dissemination

C. IoT Middleware architecture

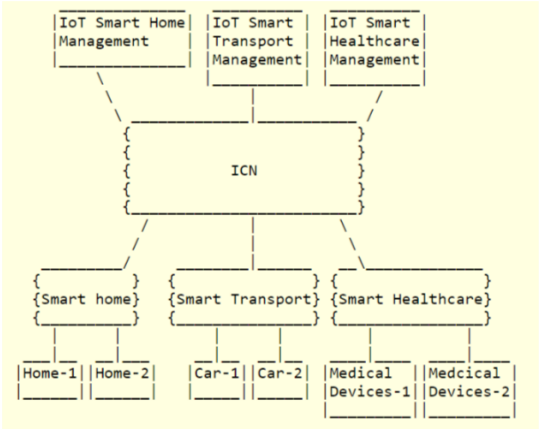

Fig. 3. ICN-IoT unified architecture

1) Advantages of using ICN for IoT: Now that we have seen that all the requirements of an IoT architecture lets see how the ICN architecture fits most of its criteria flawlessly. A key concept of ICN is the ability to name data independently from the current location at which it is stored, which simplifies caching and enables decoupling of sender and receiver.

The heterogeneity of both network equipment deployed and services offered by IoT networks leads to a large variety of data, services and devices. All these can be named exclusively by the ICN layer that can be communicate with each other through these names. These names thus lead to self configuration of devices into groups that simplify caching in strategic points of the network.

ICN advocates the model of object security to secure data in the network. This concept is based on the idea of securing information objects unlike session-based security mechanisms which secure the communication channel between a pair of nodes. ICN provides data integrity through Name-Data Integrity.

IoT networks can potentially benefit even more from caching and in-network processing systems, because of their resource constraints. Wireless bandwidth and power supply can be limited for multiple devices sharing a communication channel, and for small mobile devices powered by batteries. In this case, avoiding unnecessary transmissions with IoT devices to retrieve and distribute IoT data to multiple places is important, hence processing and storing such content in the network can save wireless bandwidth and battery power. Moreover, as for other types of networks, applications for IoT networks requiring shorter delays can benefit from local caches and services to reduce delays between content request and delivery.

IoT devices may be mobile and face intermittent network connectivity. When specific data is requested, such data can often be delivered by ICN without any consistent direct connectivity between devices. Apart from using structured caching systems as described previously, information can also be spread by forwarding data opportunistically, this allows for sender and receiver decoupling.

2) Unified IoT architecture: Now that we have looked at why ICN can be used for IoT infrastructure, we shall understand about the Middleware that will be considered for the implementation of such an infrastructure to work. The Middleware is intended to address the administrative services of this architecture such as device/service discovery, user registration and naming. The content delivery is handled by the ICN network. Thus delivering IoT services over the ICN Network. The proposed ICN-IoT unified architecture is shown in Fig.3.

The five components of the ICN-IoT architecture: are:

1) Embedded system
2) Aggregator
3) Local Service Gateway(LSG)
4) ICN-IoT Server
5) Service/Consumer

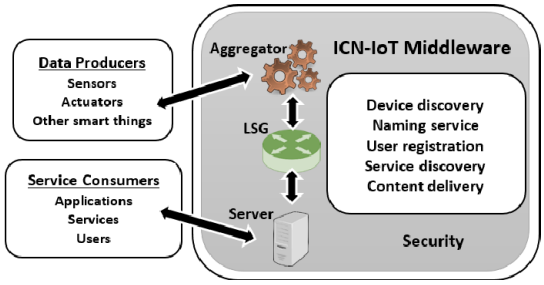

Fig. 4. Middleware architecture

This Fig. 4 shows the ICN-IoT Middleware.

Embedded Systems (ES): The embedded sensor has sensing and actuating functions and may also be able to relay data for other sensors to the Aggregator, through wireless or wired links.

Aggregator: It interconnects various entities in a local IoT network. Aggregators serve the following functionalities: device discovery, service discovery, and name assignment. Aggregators can communication with each other directly or through the local service gateway.

Local Service Gateway (LSG): A LSG serves the following functionalities: (1) it is at the administrative boundary, such as, the home or an enterprise, connecting the local IoT system to the rest of the global IoT system, (2) it serves to assign ICN names to local sensors, (3) it enforces data access policies for local IoT devices, and (4) it runs context processing services to generate information specified by application-specific contexts (instead of raw data) to the IoT server.

ICN-IoT Server: Within a given IoT service context, the IoT server is a centralized server that maintains subscription memberships and provides the lookup service for subscribers. Unlike legacy IoT servers that are involved in the data path from publishers to subscribers raising the concern of its interfaces being a bottleneck the IoT server in our architecture is only involved in the control path where publishers and subscribers exchange their names, certificates, and impose other security functions such as access control.

Services/Consumer: These are other application instances interacting with the IoT server to fetch or be notified of anything of interest within the scope of the IoT service.

D. ICN-IoT Middleware Functions

1) Onboarding: The objective of onboarding is to connect new devices to the rest and enable them to operate in the ecosystem. Every entity should be exposed to its direct upstream neighbor and may be another embedded system or aggregator. Specifically, it includes the following three aspects: a newly added ES should be exposed to its neighbor (ES or aggregator) and possibly to its LSG, AM, and the IoT server; a newly added aggregator is exposed to its LSG, and possibly to its neighbor aggregators; a newly added AM should be exposed to the IoT server and the LSG; and a newly added LSG should be exposed to the IoT server. Device discovery serves two functions: it is used in the context of discovering neighboring ESs to form routing paths, where existing mechanims can be used for device onboarding. This has been discussed in later section thoroughly.

In most IoT systems, devices interact with the aggregator for data or information processing or aggregation, hence there is no direct communication between devices under an aggregator. If in some set-up devices under the aggregator need to communicate with each other a scalable mechanism is to allow direct neighbors to communicate with each other while others communicate through the aggregator.

ICN enables flexible and context-centric device discovery which is important in IoT ecosystem where heterogeneous IoT systems belonging to different IoT services may coexist sharing the same wireless resources. Contextualization is a result of name-based networking where different IoT services can agree on unique multicast names that can be pre-provisioned in end devices and the network infrastructure using the routing control plane. This also has an advantage of localizing device discovery to regions of network relevant to an ICN service, also enabling certain level of IoT asset security by isolation. In contrast IP offers no such natural IoT service mapping; any forced mapping of this manner will entail high configuration cost both in terms of device configuration, and network control and forwarding overhead.

2) Detailed Discovery Process: A device can be an embedded device, a virtual device, a process, or a service instance such as a sensing service. We assume that the device has preloaded secure keys. Specifically, we consider both resource-constrained devices and resource-rich devices, and assume that the pre-loaded secure keys are symmetric keys or passwords for the former, while the asymmetric key pair (public key certificate and the corresponding private key) for the latter.

We assume that there is a local authentication service (AS) that performs authentication, authorization and transient action key distribution. The local authentication service is a logical entity that can be co-hosted at the LSG or IoT server. The location of the AS may be informed by efficiency, security, and trust considerations. The design offloads the complexity to the local AS and simplifies the operations at the devices. Mechanisms can be devised for authenticating and onboarding a device onto the IoT network even if the device does not trust its neighbors and the aggregator using the AS. Devices may be discovered by new device polling, mutual Authentication, Key generation and distribution or Protected Data Transfer T

3) Naming Service: The objective of the naming service is to assure that either the device or the service itself is authenticated, attempting to prevent sybil (or spoofing) attack and that the assigned name closely binds to the device (or service). Naming service assigns and authenticates ES and device names. An effective naming service should be secure, persistent, and able to support a large number of application agnostic names.

Traditional IoT systems use IP addresses as names, which are insecure and non-persistent. IP addresses also have relatively poor scalability, due to their fixed structure. Instead, ICN separates names from locators, and assigns unique and persistent names to each ES, which satisfies the above requirements. If a device needs a global unique name/ID, but does not have one, it may request the naming service to obtain one after it is authenticated. Alternatively, the IoT domain (LSG or aggregator) may determine ID (name) for an authenticated device is required based on the policy. The same naming mechanism can be used to name higher-level IoT devices such as aggregators and LSGs.

4) Service Discovery: Service discovery intends to learn IoT services that are hosted by one aggregator by its neighbor aggregators. The aggregators themselves learn service capability of the devices during the device discovery process or separately after authenticating (or during or after naming) them. The requirements for any discovery mechanism includes low protocol overhead (including low latency and low control message count), and discovery accuracy.

In today's IoT platforms, ESs, aggregators and LSGs are connected via IP multicast, which involves complicated group management and multicast name to IP translation service. Multicast, however, is greatly simplified in ICN as most ICN architectures have natural support for multicast. While it is essential that legitimate users can discover the services for which they have the proper credentials, it is also necessary that services were hidden from illegitimate users. Since service information, service provider's information, service requests,

and credentials to access services via service discovery protocols could be sensitive, it is important to keep them private.

5) Context Processing and Storage: In order to facilitate context-aware communication and data retrieval, we need to support context processing in the IoT system. The objective of context processing is to expose the ES's low-level context information to upstream aggregators and LSGs, as well as to resolve the application's high-level context requirements using lower-level ES contexts. The context processing service usually runs on both aggregators and LSGs. Context processing requires the underlying network to be able to support in-network computing at both application and network levels. ICN supports in-networking computing and caching, which thus offers unique advantages compared to traditional IP network where the support for in-network computing and caching is poor.

Once the contextual requirements and the corresponding mappings are in place, then in-network caching can be leveraged to optimize overall network performance and system QoS. In an IoT network, cached data can be used to perform data aggregation, in-network processing, and quick turnaround for answering queries. In-network caching can also be used to store data that may be relevant to many nodes and has temporal popularity in a region, and the processed data to serve the users. The contextual requirements can help define in-network processing. This goes beyond the traditional way of aggregators doing the data gathering, processing, and reduction, but also moving computation to the data (also termed network functions). Network functions in essence serves to move the computation into the network for it to happen where the context and the information is available, with the results returned to the requester. In an ICN-IoT these functionalities can be easily incorporated at scale.

6) Security: This spans across all the middleware functions. Generally speaking, the security objective is to assure that the device that connects to the network should be authenticated, the provided services are authenticated and the data generated (through sensing or actuating) by both devices and services can be authenticated and kept privacy (if needed). To be specific, we consider the approach to secure device discovery, naming service and service discovery, because other services, such as pub/sub management and context processing and storage, can be properly secured according to application-specific demands.

III. Secure Functionalities

A. Secure device Discovery

Device discovery serves two functions: 1) it is used in the context of discovering neighboring embedded systems to form routing paths, where existing mechanisms can be used and 2) for device on-boarding. During on-boarding, the Embedded system passes its device-level information (such as manufacturer-ID and model number) and application-level information (such as service type and data type) to the upstream devices. But, if the device is required to have a globally unique ICN ID, it can be provided one by the naming service.

In the name-based ICN approaches where device identity is not needed, a device can publish within the name scope of the aggregator. In most IoT systems, devices interact with the aggregator for data or information processing or aggregation, hence there is no direct communication between devices under an aggregator. If in some set-up devices under the aggregator need to communicate with each other a scalable mechanism is to allow direct neighbors to communicate with each other while others communicate through the aggregator.

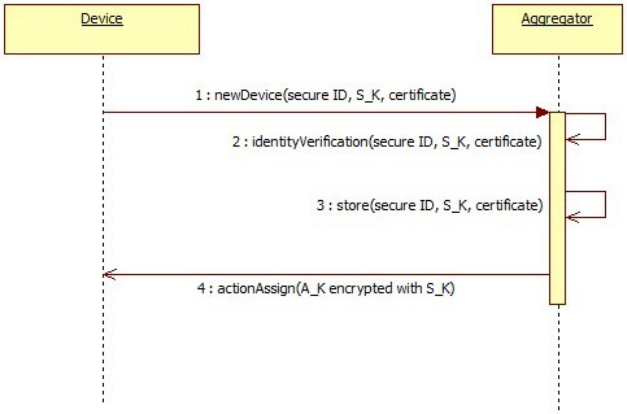

Fig. 5. Secure device Discovery with pre-load secure keys

Fig. 6. Secure device Discovery

Based on our architecture model, we would discuss, what are the different functions of the elements of this ICN system that infer secure functionalities to out IoT devices. As discussed, the aggregator is the direct connection to other devices and we shall see its involvement for these functions.

As and when a new IoT device is in the vicinity of the aggregator, it will pass on its device level information such as its manufacture ID or model number and its application level information such as its service type or data-type to the aggregator. The aggregator would then forward this information to the device that assigns unique names and returns this assignment back to the device. In doing so, it also stores this mapping of name and device for future communication.

But how is this secure? To answer this question, we shall look at the UML diagram Fig. 5, where device discovery

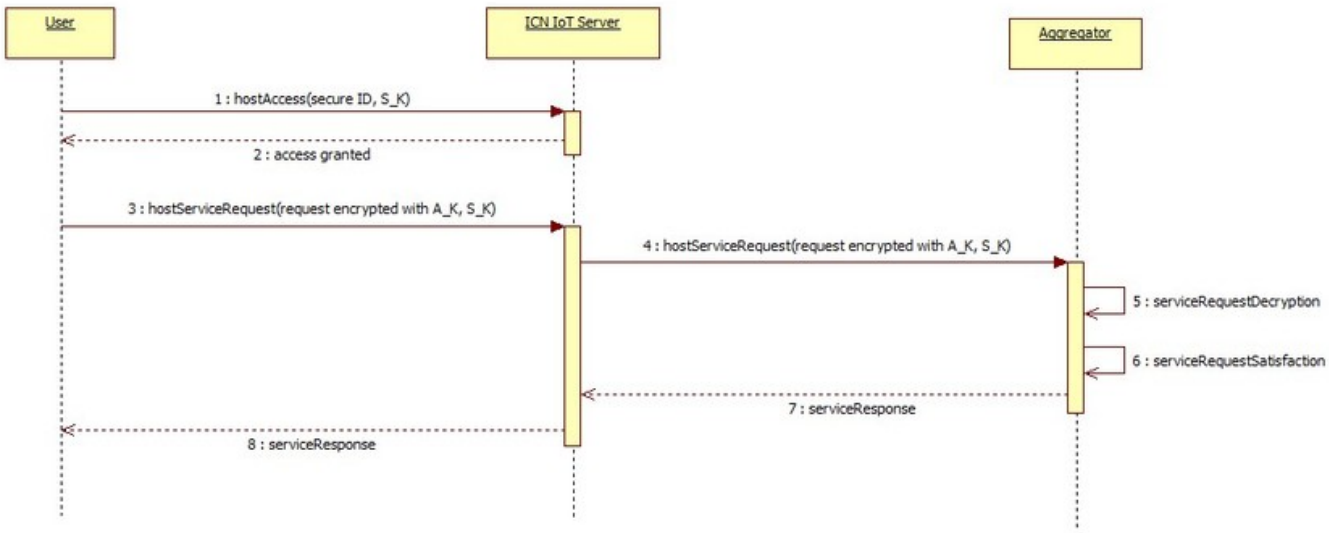

Fig. 7. Secure service Discovery

takes place due to pre-loaded secure keys. In this scenario, the embedded system is programmed by its owner with a secure ID, a public/private key pair and certificate. This information is then transmitted to the aggregator. The aggregator implements the access control mechanism and verifies the device identity and it assigns it according to the access permission policies. In order to achieve integrity and confidentiality, the device does not send its private key, but a signature derived from its private key. This is sent to the aggregator along with its certificate. The aggregator returns an action key to this device after verification. This action key is encrypted by the signature received from the device. This action key is henceforth used by the device for secure communication of future data transmissions.

Secure device discovery can also occur when the embedded system is not programmed by the owner but by the manufacturer itself. In this case, the manufacture ID , a public private key pair and its certificate will already be available on the device at the time of purchase. This method is less robust but by implementing a web-of-trust model or by ensuring that the same manufacture has provided the devices, we can trust the authenticity of the devices. This however makes us dependent on the manufacture and hence an implementation of three factor authentication by user after obtaining the device could be a round about. The same principle works as above. This can be inferred through the UML diagram Fig. 6.

First, the device sends its secure Manufacture ID to inform that it is in the vicinity. The aggregator responds with sending it its signature key and certificate. Here we must implement the three way authentication. The device will now send the secure ID encrypted with the signature which will be verified by the aggregator. In response to a positive verification, the action key will be returned to the device. This action key would be used in future secure communication between the device and the aggregator.

B. Secure service Discovery

As important as it is to secure the system for illegitimate devices, it is equally important that any illegitimate user does not access services that were intended to be hidden to this user. In order to distinguish which embedded device or user is allowed to access which services ,certain access control policies are implemented through names assigned to these devices. Let us look at the UML diagram for secure service discovery in Fig. 7.

Here, we can see the actions taken by a legitimate user that requests a certain service from the network. First, the user/device would ask the ICN-IoT Server permission to access the service. If the user is allowed to access this service then, the server permits an access grant on this device. The device now sends out a service request to the server in an encrypted manner. The encryption takes place with the action key that was earlier alloted to it by the aggregator. To maintain confidentiality and integrity this encrypted message is signed with the signature key. The server forwards this encrypted request to the respective aggregator that is responsible for this service. There, the aggregator will first decrypt the request and if it can, it will satisfy the request and send it on to the server. The server will simply forward this response to the requesting device. This works brilliantly in a distributed IoT system but as it is obvious from this example. We require a server that implements the access control mechanisms.

C. Secure naming service

If a device needs a global unique name/ID, but does not have one, it may request the naming service to obtain one after it is authenticated. Alternatively, the IoT domain (LSG or aggregator) may determine ID (name) for an authenticated device, which is required based on the access control policy. The proposed naming process works as follows. After a device has been authenticated, it may request an ID from the naming

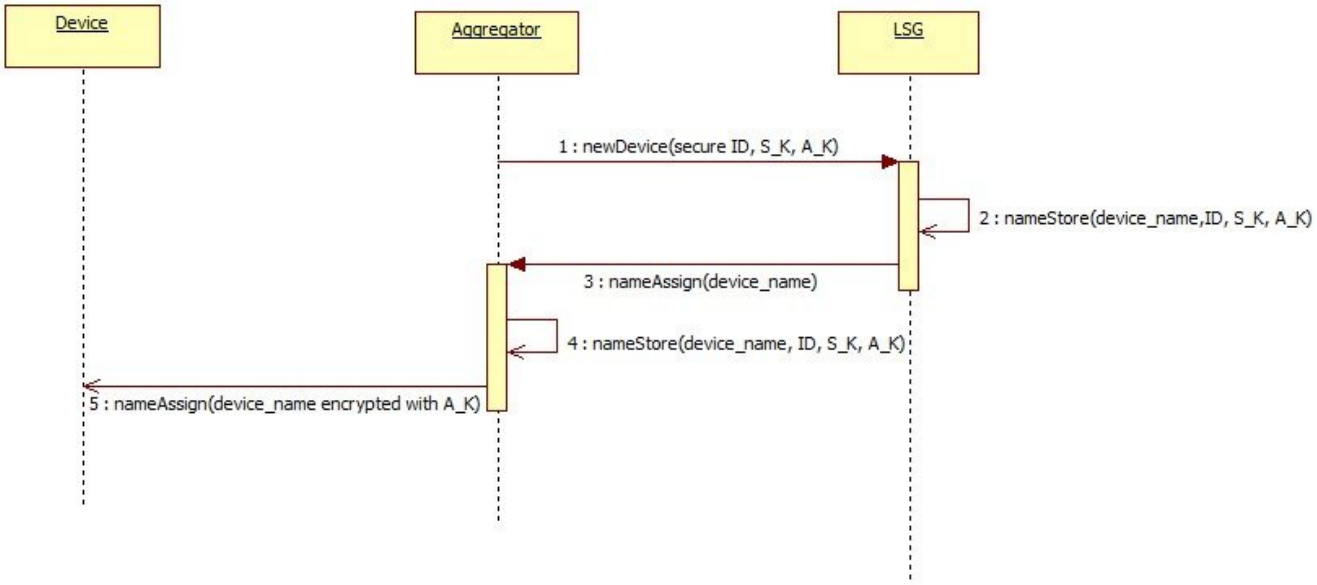

Fig. 8. Secure naming service

service (or the aggregator, if it can give the device a locally unique name). It sends a ID request (IDReq) to the naming service or aggregator. If the aggregator can accept request to give a unique name to the device, it will do that.

In Fig. 8, we see the UML diagram for naming service. As and when a new device is detected, the aggregator forwards its secure ID, signature ID and action key to the Local service gateway(LSG), which is responsible for name allocation. The LSG stores these values and finds a suitable ICN name based on its access policies for this device. Along with this name it also generates a certificate that binds the name to its signature key. This certificate infers authenticity to the device. Once a name is generated it is sent to the aggregator along with its certificate, which in turn is stored at the aggregator. The name assigned by the LSG is encrypted with the action key and forwarded to the device.

D. User registration

For a distributed system with heterogeneous embedded IoT devices, the ICN-IoT Server provides the signature key and action key required by the device for all of their functionalities. A user must register itself in order to be discovered by the aggregator to avail any services of the network. In order to do so, the user sends relevant information about himself or herself to the ICN-IoT Server as per the requirements of this application. But how can we be sure that the details entered belong to you? i.e how do we authenticate you, this will be discussed in the later section IV discussion of this report.

In Fig. 9, we see the UML diagram for user registration. Here the user sends his information to the ICN-IoT server, the server verifies this registration and returns three values back to the user: identifier, signature key and a temporary password. The user is then invoked to change his received password. Once this password has been set, the server sends the user the action key encrypted along with its signature key. Hence, upon successful registration the IoT server securely transmits an identifier, a user signature key SK (to be used to sign messages), a user encryption key action key AK (to communicate data confidentially), and an access password to the user in an encrypted message, which is changed by the user at reception.

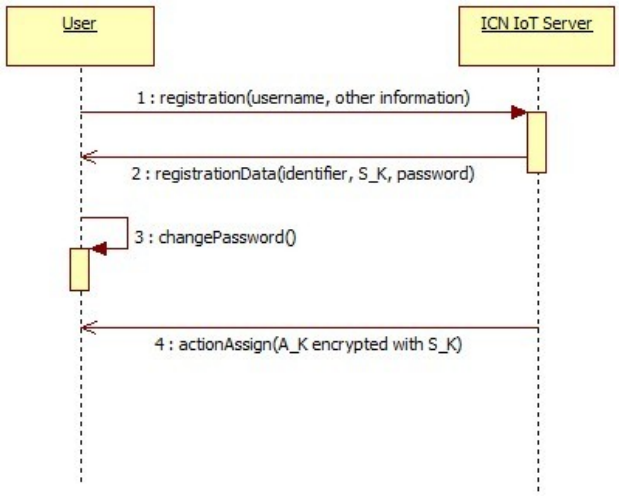

Fig. 9. User registration

E. Secure content delivery

So far we have discussed about how a user is securely registered by the system, how it is securely discovered by the network, how a new device is named according to its access policies and how a legitimate device w.r.t its name

finds a service in the network. Now we come to the last important security question: How do we secure the content that is delivered to the device from the service?

To answer this question we relay the entire scenario again. After a new device has been registered and discovered by the network, it sends its ICN name, ID encrypted by signature key and service request encrypted by action key to the aggregator. The aggregator relates the ICN name (obtained at ICN naming III-C) to map the devices action key (obtained at device discovery III-A) and decrypts the request and to verify that it is the legitimate device it decrypts the ID with the stored signature key. If the ID matches the stored device ID then the device is authenticated. Hence the request is forwarded to the server that fulfills the request and responds with the service content.

Now depending on the application, the content delivered from the server is secured with an encryption with the action key. The device must then apply decryption to read the contents and use it for its application however if this device is a light weight embedded system then it may choose not to have this encryption and use the delivered plain text content. However this is not recommended. There are other methods to increase the speed at which the delivery is made and that is through caching of data close to the legitimate devices. Another way is to use group multi-cast secure keys to relay the same information to all legitimate devices of the same group. This increases the performance of the whole system.

Fig. 10.

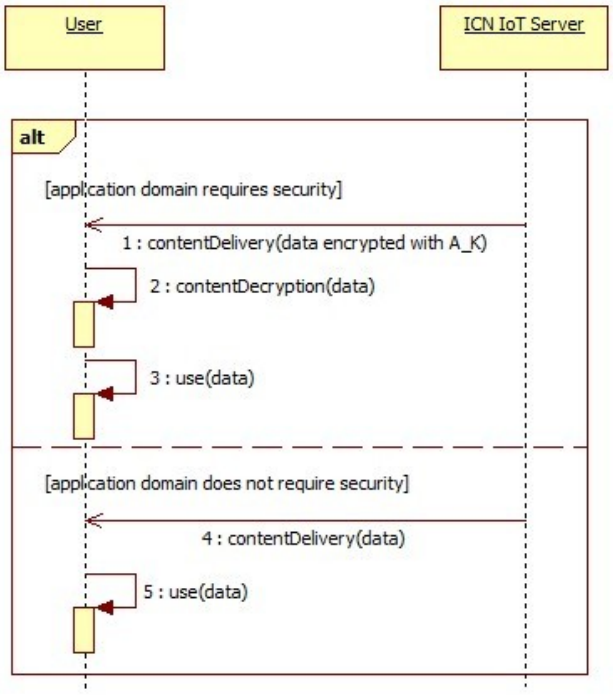

Fig. 10. Secure content delivery

IV. Discussion

There are certain open research problems in the IoT-ICN application:

1) How do applications get permission to publish and operate under a given name?
   As per our middleware, we are giving names to these devices through LSGs and aggregators but provides the permission to assign these names, is it the manufacturer of the device? In that case the system will become depended on the manufacturer and loose its heterogeneous behavior. Just as we have Internet Assigned Numbers Authority (IANA) for IP addresses , we would also require such a body to allot names. This question is still unanswered.
2) How are namespace collisions handled?
   Collisions are unavoidable whenever members of a very large set (such as all possible person names, or all possible computer files) are mapped to a relatively short bit string. This is merely an instance of the pigeonhole principle. To what degree could such namespace collision be acceptable. Or can we have a work about around it by defining scopes based on certain parameters example, geographic locations.
3) Can we break our dependence on a centralized or decentralized PKI?
   In cryptography, a web of trust is a concept used to establish the authenticity of the binding between a public key and its owner. Its decentralized trust model is an alternative to the centralized trust model of a public key infrastructure (PKI), which relies exclusively on a certificate authority (or a hierarchy of such). As with computer networks, there are many independent webs of trust, and any user (through their identity certificate) can be a part of, and a link between, multiple webs. However such a system is one of the only ways we know that ensures the authenticity of the public key. Research on this topic is also ongoing.
4) How can we adopt distributed trust models?
   One of the solutions is of trusting your bank to make a verification for you. For this trust model. The application verifies with the bank you are with to verify that you are the person, you say you are. Thereby providing a third party trust system.
5) How can we enable object encryption without sacrificing privacy ?
   With the advent of block chain technologies, this can be to a certain extend achieved. We provide privacy through the block chain and when the system is changed it ensures that your data is encrypted for all illegitimate users. However, this is not a solution for all the other web applications available. Is there a way to map the user to a trust model that does not disclose the identity of the user.
6) Can we build better broadcast encryption schemes that can handle the foreseen IoT scale?
   Broadcast encryption is a type of encryption scheme in which encrypted data is transmitted on the broadcast

channel in such a way that only privileged receivers could decrypt it. However there are always ways to decrypt these codes and thus allowing access to each and every object that was encrypted by this scheme. For the ever increasing IoT scale finding such schemes would also be a task because they exert a massive computational work which is not ideal for IoT devices.

7) How do we create secure routing hints?
   How the names are meant to create strategic points for the devices to locate different content is still under study. These depend on various naming schemes used.
8) Can we decouple "application names" from "network names"?
   Network names which are currently addressed by IP addresses are easy to distinguish from hosts and their applications but with ICN we intend to couple all naming, here too we would have to look into how such names can distinguish each other.

V. Conclusions

When the TCP/IP protocol stack was first developed in the early 1980s, the goal was to connect mainframe computers through the wired connectivity. Although the protocol stack kept evolving after the IP specification was published, the fundamental assumption behind the architecture design has not changed.

For the last 40 years, the current Internet system struggles with enabling a secure communication between hosts. IoT networks and its applications show that traditional networking is not going to work well with the functionalities required by IoT devices. For this purpose we discussed the ICN Stack which provides the IoT environment all its functionalities with security and privacy as well.

However having discussed this, traditional IP stack are with us for the foreseeable future, and ICN stack is not meant to be a replacement for it. But ICN stacks can work in conjunction to the current Internet systems ans especially the upcoming IoT systems. And therefore the ICN designs must be implemented alongside.

In this report we discussed thoroughly about the IoT requirements and how ICN provides the networkinf functionalities that it requires. We also looked into the middleware architecture that provides a good administrative structure that provides the ICN functionalities efficiently.This middleware focuses mainly on security and privacy of the overall system. Then we went on to discuss about the individual functionalities and how they provide integrity, confidentiality, authenticity and authorization. After this we discussed some of the ongoing discussions in the Internet community realted to ICNs.

ICN has been very recently addressing these security issues and still requires some time until it is fully integrated in our technology. Therefore it is of utmost importance to ensure a proper security and privacy measures to be introduced in the early stages of network design. This is one such proposal that will ensure the security functionalities required for secure communication. Eventually we would expect to see the change in the Internet paradigm and a shift towards ICN.

References

[1] Why IoT with ICN?. http://conferences2.sigcomm.org/acm-icn/2017/files/tutorial-ndn-ccnlite-riot/2-Why-ICN-for-IoT.pdf.

[2] ICN based Architecture for IoT. https://tools.ietf.org/id/draft-zhang-icnrg-icniot-architecture-01.html.

[3] Why Bother with ICN?. http://chris-wood.github.io/2016/02/05/Why-ICN.html.

[4] Design Considerations for Applying ICN to IoT. https://tools.ietf.org/id/draft-irtf-icnrg-icniot-01.html.

[5] Sabrina Sicari, Alessandra Rizzardi, Luigi Alfredo Grieco, Alberto Coen-Porisini. A Secure ICN-IoT Architecture. *IEEE International Conference on Communications Workshops (ICC Workshops)*, 2017.

[6] Wentao Shang,Yingdi Yu,Annelie Heuser,Ralph Droms. Challenges in IoT Networking via TCP/IP Architecture. *NDN, Technical Report NDN-0038*, 2016.

[7] Emmanuel Baccelli,Christian Mehlis,Oliver Hahm,Thomas C. Schmidt,Matthias Wählisch. Information centric networking in the IoT: experiments with NDN in the wild. *Proceedings of the 1st ACM Conference on Information-Centric Networking*, 2014.