Device Identity and Trust in IoT-sphere Forsaking Cryptography

Indrajit Ray, Diptendu M. Kar, Jordan Peterson
Department of Computer Science
Colorado State University
Fort Collins, CO, USA
{Indrajit.Ray, Diptendu.Kar, Jordan.Peterson}@colostate.edu

Steve Goeringer
Security Group
CableLabs
Louisville, CO, USA
S.Goeringer@cablelabs.com

***Abstract*—With the exponential growth of the Internet of Things (IoT) ecosystem there is increasing concern regarding how to ensure its security. This is particularly critical because in this ecosystem a significant number of devices are of very low computational capabilities making them particularly vulnerable to attacks. Moreover, cryptographic techniques that are traditionally used for establishing trust in entities through identification and authentication also do not appear to be suitable because of computational requirements as well as scalability issues. We present a new vision for security in this ecosystem that does not rely on cryptographic techniques and yet is able to achieve strong device identity. We also present the outlines of a crypto key less trust ecosystem that can be used to implement fine-grained access control in a pragmatic manner.**

***Index Terms*—Internet of Things, security, identity, trust**

I. Introduction

The Internet-of-Things (IoT) industry is rapidly growing with a multitude of device manufacturers releasing new smart devices (or device components that can be added to other devices) everyday to enhance our standard of living. The number of IoT device in use at the end of the first half of 2018 was estimated to be 7 billion and it is expected to reach 10 billion by 2020 ([1]). In this connected world, devices can not only be controlled by the user remotely, but one device can control another device without having the user to intervene at all. Unfortunately, security is an after-thought in a significant number of these IoT devices resulting in vulnerabilities that have been successfully exploited to cause significant damage. A widely reported example of such an attack is the Mirai IoT botnet [2] that left much of the Internet on the U.S. east coast inaccessible to users.

Several steps need to be taken to have confidence that the IoT world is significantly better protected against security attacks than what it is today. These include, but not necessarily limited to, (i) designing IoT firmware, operating system, and application so that they are (ideally) free of software vulnerabilities that can be exploited, (ii) designing robust identification and/or authentication schemes that will ensure only trusted devices are securely bootstrapped onto the network as well as allow more robust monitoring and auditing of these devices, (iii) strongly enforced access control schemes that ensure that devices have access to only those resources that they need to provide needed services, and (iv) as appropriate, cryptographic techniques to better protect sensitive communication.

Unfortunately, there is often a tendency to rely too much on cryptographic mechanisms to achieve security. For example, the Open Connectivity Foundation (OCF - https://openconnectivity.org) which has developed the OCF Security Specification for the IoT ecosystem [3], assumes that if a devices is successfully authenticated during onboarding it can be trusted for the rest of its connection session, and uses the cryptographic protocol DTLS to achieve authentication. While cryptography is an important tool in security, there are several operational issues which lead us to believe that we need to look beyond cryptography to achieve security. To begin with, many of the cryptographic techniques in use today especially public key cryptographic techniques, are computationally expensive and is beyond the capabilities of a significantly large class of IoT devices. In addition, under current public key infrastructure (PKI) market models, it is very expensive for a device manufacturer to purchase public key certificates. Many IoT device manufacturers who operate on thin margins find such costs as too expensive and choose not to use cryptographic keys for security. To complicate things, establishing and managing a PKI is very challenging. it has been observed ([4]) that a significantly large number of current PKIs are implemented incorrectly and have fatal flaws that render any security achieved under these PKIs useless in practice. In other words, cryptographic techniques often given a false sense of security.

In this work, we present a vision for an IoT ecosystem that achieves considerable practical security without using cryptography. It is based on creating unforgeable identities of IoT devices through behavioral fingerprinting that is strongly tied to the device, and on a trust framework for IoT devices that is not reputation based (unlike other trust models) but uses measurable and quantifiable properties of IoT devices to establish trust levels for them. We outline the architecture of the trust framework as well an access control model that uses

This work is partially supported by CableLabs. This material is also based upon work performed by Indrajit Ray while serving at the National Science Foundation. Research findings presented here and opinions expressed are solely those of the authors and in no way represent the opinions of CableLabs or the U.S. NSF or any other federal agencies.

978-1-7281-6741-1/19/$31.00 ©2019 IEEE
DOI 10.1109/CIC48465.2019.00034

the underlying trust model.

II. Towards Unique Device Identity

Many security problems can be mitigated through strong identification and authentication of devices, which enables administrators to effectively monitor devices and enforce appropriate access controls on a particular device. Proper identification needs the device to be associated with an unforgeable identity that is also stringly tied to the devices. We posit that such an identity can be established by appropriately characterizing the behavior of the device through behavioral fingerprinting. The latter also helps establish a behavioral baseline that answer the question "Is the device doing what it is supposed to do." For example, a light bulb mostly performs three tasks - turn on, turn off, and adjust brightness. Scanning the network is not its desired behavior.

Fingerprinting IoT devices is challenging due to the large variety of devices, protocols,and control interfaces, across the devices. Fingerprinting involves modelling the normal behavior based on network traffic features that characterizes a device. An untrusted device can masquerade as another device by providing false information about its identity and type. More importantly, an untrusted or compromised IoT device might behave contrary to its baseline behavior, e.g., connecting to other devices to disrupt their normal functionality or to scan the network for information about other devices.

IoT device type fingerprinting research is in early stages due to the evolving nature of the IoT industry. General device fingerprinting has been described in [5], [6], [7], which have explored several techniques ranging from packet header features to physical features such as clockskews. Wireless device finger printing techniques have been discussed in [8], [9], [10], [11], [12]. These works explored the device type identification by exploring the implementation differences of a common protocol such as SIP, across similar devices. However, IoT devices use numerous protocols and it would be nearly impossible to attempt such analysis on a per protocol and per device basis. Physical layer based device fingerprinting has received considerable attention [13], [14], [15], [16], [17] where the focus is on analyzing the physical aspects of devices to fingerprint them. All these works focused on general wireless devices and their applicability to IoT devices is an open question.

Miettinen et al. [18] described IoT Sentinel, a framework for device fingerprinting and securing IoT networks. Their work focused on supervised machine learning techniques for fingerprinting a device when it first registers on a network. Bezawada et. al. [19] described IoTSense. This work is focused towards the active state of the device i.e. what the device is doing on the network. They also use supervised machine learning techniques to characterize the behavior of the device on the network. Both of these works, IoT Sentinel [18] and IoTSense [18], focus on identifying a device of a specific type; however both suffer from the problem that if a device of an unknown type joins the network it is wrongly classified as one of the known devices which the algorithms have learned before.

Marchal, et al. [20] overcomes the shortcomings of IoT-Sentinel to propose AuDI that is able to detect an unknown devices type with a high degree of accuracy.However, this method relies heavily on low network congestion. Because the method is dealing with flow frequency, if there is any network congestion this frequency has the potential to be misaligned with the device fingerprints. Another issue with this approach is that it takes 2.5 hours to train a new device type, and 30 minutes to collect enough idle traffic to classify a device with such high accuracy.

We now describe our initial efforts in developing an unsupervised learning approach for fingerprinting devices. We describe our approach on device fingerprinting at the device bootstrap or onboarding phase, when it first registers on a network. However, this approach can also be used to create device fingerprint both when the device is active and when it is idle. Our results are very encouraging and we are convinced that this approach can be extended to create unique fingerprints of devices that can be used as the basis of a strong device identity. When paired with fingerprints of devices at the idle and active states, this approach can form the basis of an effective monitoring system for IoT devices. We have tested our approach not only on different device types (at the level of granularity such as music systems, lights, camera, voice assistants etc. but also on similar devices. For example, we have trained on Amazon Echo dot (smaller one) and Echo (bigger one) and found that we can distinguish them distinctly. Neither the IoTSentinel approach nor the IoTSense approach can lay claim to this particular feature and it remains to be seen if AuDI can.

III. Proposed Learning Model

The model we are proposing is an unsupervised learning model that can identify devices based solely on network traffic. Our model doesn't take into account any traditional addressing techniques such as IP addressing or MAC addressing. In this section we describe the model features and classification technique used.

A. Model Features

As a passive identification model we observe four types of messages: DNS Query, DNS Response, SSL Cert, Client Hello. These message types were chosen due to their ubiquity between home IoT devices. From each of these message types we mined common and specific features.

Common: The common features gathered from each message were frame length, ip length, tcp/udp length. The frame length in this context is the size of the whole packet. The ip length is the size of the ip packet, and the tcp/udp length is the size of either the tcp portion of the packet or udp portion of the packet depending on which protocol the message type uses. For example the DNS Query and DNS Response will use udp length, the SSL Cert and Client Hello will use tcp length.

Specific: Each message type has specific data that is mined. The features are listed below by message type.

- **DNS Query:** query name length, url score.
- **DNS Response:** response name length, url score.
- **SSL Cert:** certificate length.
- **Client Hello:** handshake length, handshake cipher suite length, handshake extensions length, handshake extensions server name length, url score.

The Url score mentioned above is a value computed based on message type and domain names. The score depends on how close the url in question matches the trained urls. For example, if the url in question is aaa.bbb.ccc and it matches exactly then the score is 100. if it does not match then we take off one sub-domain to make the example *.bbb.ccc. After this we check and continue this process until there is only the core domain left. If there was a match the score is set to be slightly lower at 99. If there was no match then the score is set to zero.

B. Classification Method

For each device in the training phase, four clusters are created based on the message types described above. Similar message type clusters are aggregated to form a model which is used for evaluation of that specific message type. The centroids of each cluster are stored and used to determine how far the test data sample is from the centroid. A threshold value of 100 is provided. This can be set by the user for a tighter or relaxed evaluation constraint. As an example, if there are 4 devices in training d1, d2, d3 and d4, the first model contains the dns request cluster centroids of d1 to d4. The second model contains the dns response and so on. During evaluation when a device d_n connects to the network, it's communication messages are grouped by message type, sent to the four models and the cluster number to which it matches is returned. There is often a case that a single message may return match with more than one cluster. Hence, the final prediction is aggregated and displayed after a certain time interval. Also the scores from each model is taken into account and the final score is an average of the four models. This is to verify that all types of messages indicate that they are from the same type of device and there might be a possibility that one type of message will be same for two different devices. The algorithm that was used for generating the clusters was K-means. Any other clustering algorithm will work as well and the accuracy might improve. We have not tested with other unsupervised clustering algorithm as our goal relies on using any unsupervised algorithm rather than a supervised one.

IV. Experimentation and Results

For this work we performed two experiments. The first experiment was to see if our classifier could correctly distinguish device boot-up traffic from trained and unknown devices. The second experiment was to validate our claims tested by the first experiment. For both of the experiments below we developed a testing script in Python to collect network data and pass it through the classifier. The two experiments included the IoT devices listed below.

The first experiment that we performed aimed to test the correctness of our learning model. In order to examine if the features that we chose were significant, we tested against 13 devices listed in table I. Our model generated eight labels that represent the eight device that we used to train the classifier. The ninth label (99) was reserved for any device that the classifier deemed as unknown. This is the default label for any device which was not used in training and not classified within the bounds of any trained centroid during testing.

TABLE I
List of devices used in experiments

Device Name	Label	Manufacturer	Interface
Fire Stick TV	0	Amazon	WiFi
Apple TV 1	1	Apple	WiFi
Apple TV 2	2	Apple	WiFi
Cloud Camera	3	DLink	Ethernet
Echo	4	Amazon	WiFi
Echo Dot	5	Amazon	WiFi
ArloQ Camera	6	Netgear	WiFi
TPLink Bulb	7	TPLink	WiFi
Wink Hub	99	Wink	Ethernet
Echo Show	99	Amazon	WiFi
Home Mini	99	Google	WiFi
Xperia Phone	99	Sony	WiFi
Nexus 5e Phone	99	LG	WiFi

Our results for this experiment showed that of the 14 devices tested, all but one was classified correctly. The details of the classification probability for two of our unknown devices are shown in Tables II and III. The total number of packets that the boot phase generated are in the total row. Out of that the number of packets matching the labels are in each column. From table II it can be observed that during the boot phase of Google Home, there were 53 DNS requests. Out of 53, 35 suggested that the request was similar to Amazon Firestick (label 0) and 18 are never seen before (label 99). So just by looking at the DNS requests, the device is likely to be an Amazon Firestick. But when the rest of the three models are consulted, the final probability suggests that the device is 99 - not matching with any cluster or unknown. The one device that was misclassified was the Amazon Echo Show. This device is very similar to the Echo and Echo Dot. Below in Table III you can see the test results for the misclassified device.

Our second experiment aimed to validate the correctness of our learning model and provide insights into how we can improve the model.

For this experiment out of the nine devices tested eight were correctly classified by the end of boot up process. The one device that was misclassified was again the Amazon Echo Show. This is no surprise as we did not alter the training of the classifier between the two experiments. Looking at the probabilities and confidence that we calculated, we found that model did correctly classify the Amazon Echo Show as an Amazon product. In Figure 1 we see that the device with the highest ending probability was the Amazon Firestick with 74.26%. The classifier correctly identified the Amazon Echo Show as an Amazon product through the calculated confidence between the two devices but it was technically wrong as it should have predicted 99 or unknown. This makes intuitive sense as these Amazon devices are very similar in behavior and

functionality. In Figure 1 and Figure 2, we see the probability and confidence of the misclassified device over time.

These results show that untrained on the Amazon Echo Show's boot data the classifier performs very well when identifying devices at coarse granularity, i.e. brand, type, etc. In order for the classifier to identify the device at fine granularity such as the specific model of device, Echo dot vs. Echo show, we needed to add the cluster of the new device with the boot data of that device to the existing clusters. Once we added the new cluster we achieved a final probability of 90.4%. Seen in Figure 3 the classifier has now correctly identified the Amazon Echo Show. In Figure V, the classifiers highest ending probability and peak confidence results for each devices boot data are listed. A new label 8 was added to mark the centroid created when training the Amazon Echo Show.

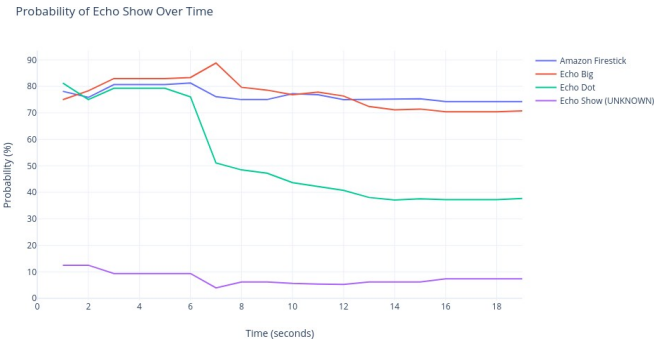

Fig. 1. Probability of Echo Show Over Time

Fig. 2. Confidence of Echo Show Over Time

V. Trust Model for IoT Ecosystem

The main motivation behind the trust model is to implement the ability to enforce fine-grained access control on an IoT device depending on how its different properties and behavior are evaluated. There are several trust models that have been proposed over the years mostly reputation oriented.

The model enables on to compute device trust values, which are normalized values in the range [0, 1]. We discuss in this section the proposed IoT ecosystem trust model, which can

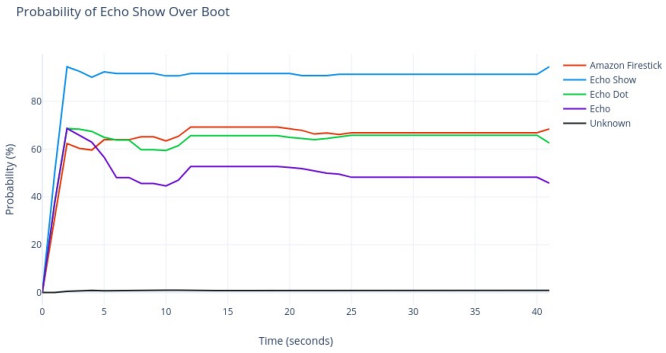

Fig. 3. Probability of Echo Show Over Time: Trained

estimate the trust of various devices based on some metrics. The trust model computes the degree to which an attached device can be trusted by measuring different trust indicators relevant to the device and monitoring the trust triggers that change values of these trust indicators. The trust values at any given period of time are used to enforce the access control policies corresponding to that trust levels.

A pre-requisite for using this trust model is the ability to associate a unique identity to an IoT device such that the identity is neither forgeable nor can it be dissociated from the device. The current device trust value, previous trust values and, more importantly, the evaluated parameters used to compute the trust values are stored indexed by this device identity. In Section II we discussed our initial attempts to generate unique identifier of devices through behavioral fingerprinting.

Before presenting the IoT ecosystem trust model, we give a high-level overview of the trust framework architecture with the various components and their interactions in Figure 4. The framework consists of the *Trust Management System* which has two components – the *Trust Model* and *Trust Evaluation Engine* – the *Trust Based Access Control System* (discussed later), the *Device Monitoring System* that monitors trust triggers and evaluates the trust indicators relevant to devices, and the *Access Decision Point* which is any device or process working on behalf of an entity that needs to determine whether further access can be permitted to the requesting device. The interactions between these components are indicated by directed and labeled arrows in Figure 4. The number against a label represent the ordering of interactions in an access control decision.

At system initiation time, the Device Monitoring System learns the identify of an IoT device via the fingerprinting approach discussed in Section II. This behavior-based identity is converted to a unique string representation and shared with the Trust Management System for indexing of device trust values. Subsequently, the Device Monitoring System continuously monitors the devices in the system to ensure that only a known device (that is a device whose identity is available) exists in the system and evaluates various trust

TABLE II
EXPERIMENT 1: RESULTS FOR GOOGLE HOME

Prediction label	DNS Query	DNS Response	SSL Cert	Client Hello	Probability
0	35	8	0	0	0.2402
99	18	16	17	23	0.7621
TOTAL	53	24	17	23	

TABLE III
EXPERIMENT 1: RESULTS FOR ECHO SHOW

Prediction Label	DNS Query	DNS Response	SSL Cert	Client Hello	Probability
0	177	174	190	61	0.71272
4	170	176	129	73	0.65659
5	141	142	76	42	0.48155
99	141	142	76	42	0.16718
TOTAL	206	206	231	188	

TABLE IV
EXPERIMENT 1: RESULTS FOR AMAZON ECHO

Prediction label	DNS Query	DNS Response	SSL Cert	Client Hello	Probability
0	25	24	18	20	0.5972
4	34	33	15	24	0.75
5	32	31	17	21	0.5972
99	0	0	6	0	0.1250
TOTAL	34	33	24	24	

TABLE V
EXPERIMENT 2: PROBABILITY AND CONFIDENCE RESULTS

Predicted Label	Probability	Confidence
0 (Amazon Firestick)	83.05 [0]	95 [0]
1 (Apple TV 1)	59.2 [1]	85.71 [1]
2 (Apple TV 2)	57.3 [2]	80.32 [2]
3 (Dlink Cloud Camera)	100 [3]	100 [3]
4 (Amazon Echo)	75 [4]	96.4 [4]
5 (Amazon Echo dot)	76.44 [5]	90 [5]
7 (TPlink Bulb)	100 [7]	100[7]
8 (Amazon Echo show)	90.4 [8]	87.5 [8]

indicators and updates these to the Trust Management System. These steps are indicated by "0" in the figure.

When an access request is made by a device (arrow labeled "1"), the Access Decision Point evaluates the device trust by interacting with the Trust Management System (arrow labeled "2"). The request for an access triggers the Device Monitoring System to push the latest update of the device's trust indicators to the Trust Management System (arrow labeled "3"). The computed device trust value is then sent over to the Trust-Based Access Control System (arrow labeled "4"), which returns a set of allowed accesses (arrow labeled "5"). Finally, the Access Decision Point makes a decision to allow or deny the access (arrow labeled "6"). This decision is also recorded by the Trust Management System (arrow labeled "7") and is used in subsequent evaluation of device trust.

Although the job of the Access Decision Point is similar to that of the Policy Enforcement Point of XACML, we envision it to be the conduit through which the IoT devices interacts with the rest of the world. For example, the Access Decision Point can be the cable modem router through which a smart home IoT device connects to the cable broadband ecosystem or it can a IoT hub that connects a multitude of non TCP/IP speaking IoT devices to a TCP/IP network, and so on. Thus, in our model the Access Decision Point is the trustor and the IoT device is the trustee. For the rest of the discussion we will refer to the Access Decision Point as the trustor to which a device connects and use the symbol *CM* to refer to it.

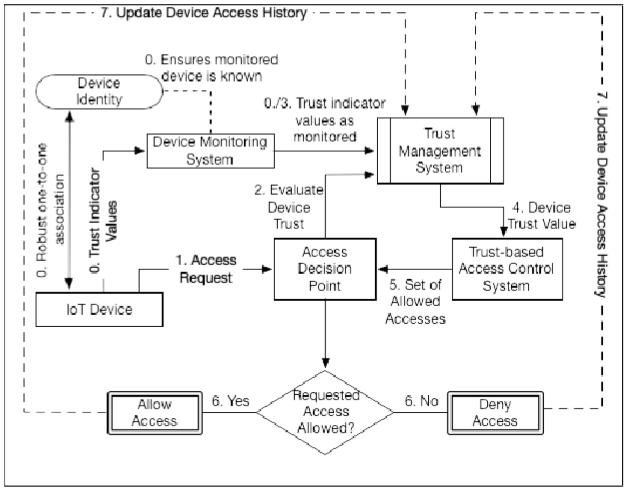

Fig. 4. IoT Trust Framework

A. Trust model components

The ADP system trusts a device connected to it to different degrees for different access types. An access type intuitively captures the nature of resource-access requested by a device. Note that, our definition of trust is not absolute or specific

to a device but depends on the context of the access that is captured by the notion of access type. Context may include architectural, topological, internal and external security indicators, and other factors that may change in time (in other words, context varies temporally). We model the access types of the devices using an access graph, which is described at a later sub-section. In the following, we describe the outline of the trust model formalism and the intuitions that operate within the model.

A. *Access type based trust*: We use the expression $CM_i \overset{c}{=} D_j$ to denote that trustor CM_i trusts device D_j (the trustee) for a specific access type c. The access type determines what specific actions are possible on various resources under control of CM_i.

B. *Degree of trust*: A truster trusts a device to a specific level. This degree of trust is computed periodically or is triggered by the occurrence of one or more *trust triggers*. These trust triggers are events that change the value of a trust indicator. The expression $V_t(CM_i \overset{c}{=} D_j) \in [0 \ldots 1]$ represents the degree (or level) of trust that the trustor (also referred to as the trust evaluator) has on the trustee for the access type c as computed at time t.

   a. *Lack of trust* - A value of 0 means that the trust evaluator has no trust on the device. This can be because the evaluator does not have any information about the device to compute a trust value or that the result of trust evaluation has been 0.

   b. *Fully trusted* - A value of 1 means that the trust evaluator fully trusts the device.

C. *Trust history*: The trust management system corresponding to a trustor maintains a history of trust values determined for the trustees it has encountered. The history also contains a log of the trust indicator values that resulted in the corresponding trust levels of the device. The trust history allows one to keep track of how the trustworthiness of a device has evolved, if at all, over time and plays a role in determining the current trustworthiness of the device. The rationale behind allowing the trust history to affect the current trust level are two: (a) There may be situations when the relevant trust indicators cannot be evaluated for various reasons; a trust history allows a reasonable evaluation of trustworthiness under such circumstances. shows a scenario where such trust history is useful. (b) A device that used to be trusted (at some level) at some time cannot become overnight un-trusted. A trust history score allows arbitrary changes to be smoothened out.

D. *Trust history log*: For every trustee that the trustor has ever evaluated, the trust history log is an update only database of $(key, value)$ pairs, where key is the time instance when the trust value is being computed and recorded and $value = [P_{dev}, h_t]$, where P_{dev} is the trust indicator score of the device and h_t is the trust history score of the device computed at current time, t_n, using the trust level of the device at time t_{n-1}.

E. *Trust history score computation*: The trust history score h_t at the current time t_n is based on the trust value $V_{t_{n-1}}$ computed at time t_{n-1}. Let $\Delta t = t_n - t_{n-1}$. Then, the trust history score at time t_n is given by $h_t = V_{t_{n-1}} \times e^{-((V_{t_{n-1}})^{-1}\Delta t)^{2k}}$, where $k \geq 1$, is a small integer that determines the rate of change of trust and is assigned by the system as a system policy.

F. *Trust value at current time*: The degree of trust that a trustor has on the trustee at the current time is given by $V_{t_n}(CM_i \overset{c}{\Rightarrow} D_j) = P_{dev} \otimes h_t$, where $\otimes$ is an operator defined in a later section for combining the two trust parameters into a single value in $[0 \ldots 1]$.

B. Trust model defintions and parameters

The IoT trust model defines device trust indicators to quantify trust and a belief system to determine accuracy of trust indicators. Each device is associated with a vector of trust indicators, which is used to determine the trust indicator score P_{dev} that is finally used to evaluate the trust of a device.

DEFINITION 1 A device *trust indicator* is some measurable property of the device whose value impacts the trust level of the device. An *internal* trust indicator of a device is either a property that is intrinsic to the device or a property associated with the device that emerges from within the IoT ecosystem. An *external* trust indicator, on the other hand, is either a property that is extrinsic to the device or that emerges from outside the IoT ecosystem.

DEFINITION 2 A *trust trigger* is an event that changes the value of a trust indicator. Not all trust indicators have associ- ated triggers; some may have more than one trigger while others may share trigger. Trust triggers can be internal or external.

DEFINITION 3 *Trust indicator vector*: Every device is associated with an ordered set of size m of trust indicators $p_t = [k_1, k_2, \ldots, k_j, \ldots, k_m]$. The trust indicator k_j has a value k_j . It can be evaluated by the Device Monitoring System. However, some trust indicators can be measured only probabilistically with some degree of confidence while others can be measured accurately.

DEFINITION 4 *Confidence in trust indicator value*: The confidence in trust indicator value is a measure or percentage of accuracy of a given trust indicator value.

Although all attempts are made to measure a trust indicator accurately, there is always a degree of uncertainty as to how accurate the trust indicator value is. To address these concerns, we define a belief system based on subjective logic [21] that

allows the adjustment of the trust indicator values based on the perceived beliefs of the measurements. Each time a trust indicator k_j is evaluated, the proposition "*the measurement is accurate*" is subjectively judged by associating it with four values: (i) a belief $b_{k_j} \in [0 \ldots 1]$ that the proposition is true, (ii) a belief $d_{k_j} \in [0 \ldots 1]$ that the proposition is false, (iii) a belief $u_{k_j} \in [0 \ldots 1]$ that is neither committed to the truth nor falsehood about the proposition, and (iv) an a priori probability a_{k_j} about the the truth of the proposition in the absence of a specific belief. Since this is a binary state space, a_{k_j} is set to 0.5 as per the theory of subjective logic described in [21]. The values b_j, d_j, and u_j are related as $b_j + d_j + u_j = 1$. Using these values the confidence level of a measured trust indicator is given by $o_j = b_j + a_j u_j$. As a trust indicator is being evaluated, we assume that the belief, disbelief and uncertainty values associated with the measure would also be provided. An example of computation of confidence in trust indicator value is given in Figure 5.

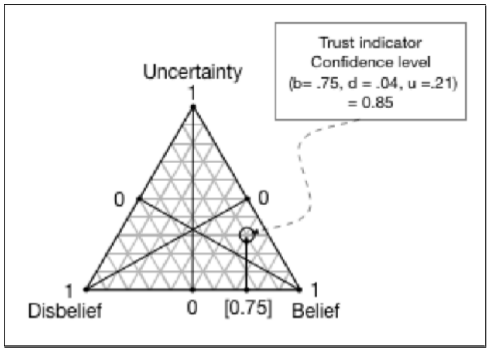

Fig. 5. Trust Indicator Confidence Level Computation

DEFINITION 5 *Trust indicator importance vector*: Device trust indicator importance is represented as a 1-dimensional vector $p_{imp} = [p_1, p_2, \ldots, p_j, \ldots, p_m]$, where p_j represents the importance weight of trust indicator k_j.

- *Trust indicator score*: The trust indicator score P_{dev} of a device at time t_n is a single value in the range $[0 \ldots 1]$ computed as a weighted average over the values in the trust indicator vector using the values in the trust indicator importance vector and the corresponding confidence values.

The trust indicator score is computed as follows:

1) Define a single value function

$$f_j(k_j) = \frac{|k_j|}{\max_j |k_j|} \times 100$$

for each trust indicator, where $|k_j|$ represents the value of the trust indicator k_j. The purpose of this function is to normalize different unit measures so that the values can be summed together under single standard scale.

2) Multiply each normalized trust indicator values in p_j with the corresponding trust indicator confidence level and trust indicator importance and then normalizing it to a scale of $[0 \ldots 1]$.

3) Compute the trust indicator score for the device as:

$$P_{dev} = \sum_{j=1}^{m} \frac{f_j(k_j)\, o_j p_j}{\max_j \{f_j(k_j)\, o_j p_j\}}$$

C. Trust indicators for device trust in smart home IoT

To use the trust model in practice one has to determine a set of suitable trust indicators that can be measured. In association with our industry partner in the broadband-over-cable industry, we undertook an investigation of what can serve as potential trust indicators in a smart home IoT environment. Below we list several device trust indicators. We have categorized them according to their functionality and not necessarily by importance. A system designer is free to choose the relative importance of the categories and the components within the categories. Trust indicators are scored on a point basis. If the existence of a trust indicator positively helps / enhances / impacts / sustains the security of smart home IoT, then a positive point is assigned. If it impacts negatively, a negative point is assigned. The trust indicators we have identified are:

A. Device category – Evaluation decision is based on what impact would a compromise of a device in that category have on the system. Impact is evaluated on a Likert scale. Various subtypes are also identified:
   i) End-point networking - Examples include IoT hubs, switches, routers, wifi access points, etc.
   ii) Productivity - Examples include tablet, smartphones, smart printers, etc.
   iii) Entertainment - Examples include smart TV, speakers, music systems, media hub etc.
   iv) Physical safety and security - Examples include motion detector, security camera, fire alarm, smoke detector, doorbell etc.
   v) Home comfort and convenience - Examples include thermostat, bulb, electric switch etc.
   vi) Utility - Examples include refrigerator, washer, dryer etc.
   vii) Medical and health care devices - Examples include smart watches, wearable health monitors etc.
   viii) Voice assistants and controllers - Examples include devices such as Echo show, Google Home etc.

B. Device capability - The higher the device's computational capabilities the more damage can probably be done by exploiting the devices. Factors considered are (i) type of processor (microcontroller, general purpose etc. (ii) available RAM, and (iii) storage capability.

C. Network connection type - wired, wireless type (WiFi, cellular, Bluetooth, Zigbee, Zwave, etc.)

D. Physical location of device - (i) Unprotected (no physical barriers), (ii) partially protected (some physical barriers to access), or (iii) high protection (inside house only accessible to owner or by authorization of owner).

E. Connection requirement - (i) device to device (ii) device to cloud, (iii) application to device or (iv) application to cloud
F. Connection security - (i) Authenticated connection (Y/N), (ii) Encrypted connection (Y/N), and if encrypted connection (i) encryption type and (ii) encryption strength.
G. Connection behavior - (i) device initiated or outside initiated, (ii) average frequency of connection with outside, (iii) average duration of connection, (iv) average bandwidth consumed per connection (v) frame rate for traffic, (vi) traffic type, (vii) signal strength (viii) connection port numbers, and (ix) packet length deviation (expected vs. observed.
H. Current configuration of device - (i) default, (ii) updated, and (iii) latest firmware with patches updated
I. Device manufacturer trust level

No that this list is, by no means, complete. Our preliminary investigations have led to the determination of this list of trust indicators. Depending on the scenario, this list may need to be updated. In addition, recall from earlier discussions, that each trust indicator is associated with a confidence level that determines one's belief about how well a measured trust indicator score is.

D. Trust management system

The trust management system is responsible for monitoring and managing device trust indicators as well evaluating trust levels of devices and answering queries related to device trust. It consists of the following components:

- Immutable database to store and manage trust related data
- Immutable log of trust history
- Trust specification engine for defining and managing trust relationships
- Trust analysis engine to process the results of a trust query
- Trust evaluation engine for evaluating trust relationships
- Trust monitor for monitoring trust triggers to update trust relationships

The architecture of the trust management system is shown in Figure. The green-colored arrows in the figure reflects communication between various modules that happen periodically in order to update trust levels of devices; the red-colored arrows represent communications that occur when the trust management system is queried for trust value related information. Finally, the black-colored arrows represent internal operation of the trust database.

VI. Flexible Trust-based Access Control for IoT

The purpose of defining the new trust model is to enable fine-grained access control of the connected devices. A factor that is often overlooked when fine-grained access decisions are made is that there can be various tradeoffs in terms of cost of effectuating permission, cost of consequence for a coarser grained versus finer grained permission etc. that need to be accounted for in determining the granularity of access. One of the novel features of this access control paradigm is it provides an opportunity of expressing policies by trading off factors. In this section, we describe the trust based access control framework that achieves this task.

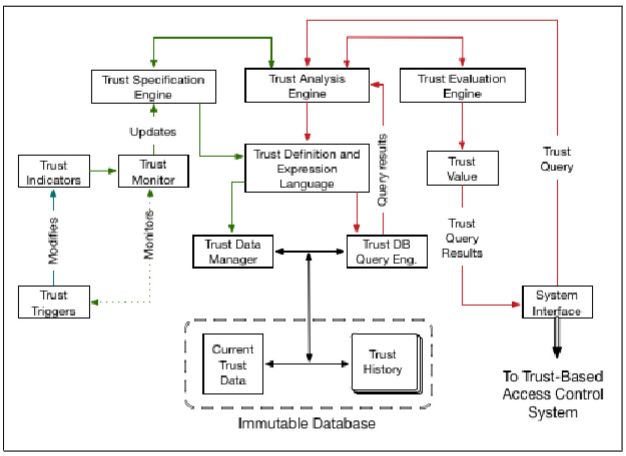

Fig. 6. Trust Management System Architecture

A. Trust-based access control model

The main elements of the access control model are the devices, the access types and the resources. Devices request access of a specific type to the system resources. An access request can be satisfied by one or more set of permissions to a set of resources managed by the system. Some key concepts in this model are:

- A permission is defined as an action on a specific resource (object).
- A permission has a *consequence cost* and *cost to effectuate* it that are determined during the time of access control policy establishment.
- An access request has an associated trust value (or range) that specifies the level of trustworthiness needed for a device to be granted that access request.

Every device has one and only one trust history; a device also has a set of trust indicators as described in the trust model. When a device requests access to resource, the trust based access control system determines:

1) Whether the device's trust level allows the desired access, i.e., is the device trust $\geq$ trust level needed for access.
2) The cost of the consequences of the set of permissions and the cost of effectuating the set.
3) The set of permissions that effectuates the access request and at the same time minimizes the costs.

The last item is important to note. The access control framework returns a set of permissions that are allowable for a particular scenario. What this allows is that if an access request involves multiple permissions but only a subset of that can be allowed, a decision can still be made to allow partial access instead of complete access denial.

1) Model elements: We refer to Figure 7 for the following discussion. It shows the different model elements and their relationships.

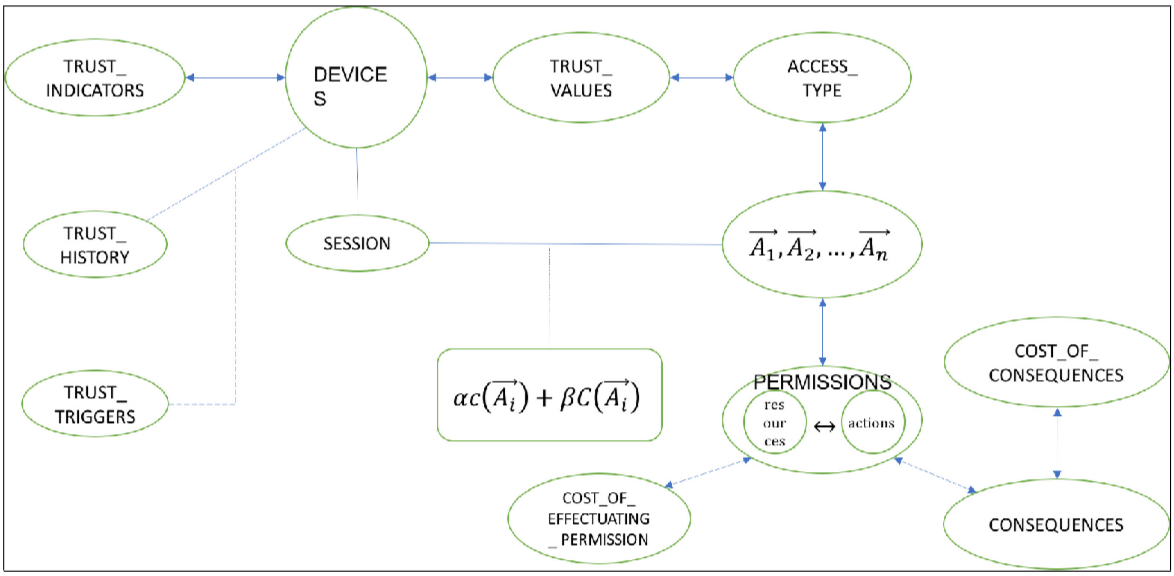

Fig. 7. Trust-based Access Control Model for IoT

The trust-based access control model is defined in terms of a set of elements and relationships among those elements. The different elements are:

i. Device: A device $\in$ DEVICES is defined as an entity that is connected to the system and requests access to some resources controlled by the system.
ii. Trust Indicators: Each device, d, has an ordered set p_d of measurable properties called trust indicators. The set TRUST INDICATORS = $\bigcup_{d \in DEVICES} p_d$. A device can manifest measurable values for any subset $p \subseteq p_d$ during a specific session. (See earlier for discussion on trust indicators.)
iii. Trust history: A trust history $\in$TRUST_ HISTORY is a set of information regarding a device's old trust values as evaluated in various times including in a previous session.
iv. Trust trigger: A trust trigger $\in$ TRUST TRIGGERS is an event that changes the value of a trust indicator or a clock tick designating a time instance where a change of value for a trust indicator has occurred. A trust trigger causes a trust history to be updated.
v. Session: A session $\in$SESSIONS is a connection instance of a device. A device requests access to system resources during a session. A device can instantiate multiple sessions. A session is identified by a unique system generated id.
vi. Trust Level: A trust level is a set of real numbers between 0 and 1. A device at some instant of time in a session has a trust level. The set TRUST LEVELS is the set of possible subsets of $[0 \ldots 1]$.
vii. Access Type: An access type specifies the nature of access that is sought. An access type, $AT_i \in$ACCESS_ TYPES is a set of keywords with some associated semantics regarding the resources needed and the outcome.
viii. Resource: A resource $o_j \in$ RESOURCES is an object controlled by the system that needs to be protected.
ix. Actions: An action $x_i \in$ACTIONS is an operation that can be performed on a resource o_j.
x. Permission: A permission $a_k \in$ PERMISSIONS is an authorization to allow certain action. It is defined as an element of RESOURCES $\times$ ACTIONS, i.e., PERMISSIONS = $2^{\text{RESOURCES} \times \text{ACTIONS}}$.
xi. Consequence: A consequence $\xi \in$ CONSEQUENCES is the resulting system state when a permission is effectuated.
xii. Access: An access A_i is an ordered subset of PERMISSIONS. Every access is associated with an access type.
xiii. Cost of effectuating permission: A cost of effectuating a permission is a real number from a finite, discrete set of real numbers ≥ 0 that represent an unitless cost amount incurred by the system in enabling the permission.
xiv. Cost of consequence: When a permission is effectuated, it results in a change of system state which is the consequence. The changed system state can be better than, worse than or same as the original state in terms of security. Regardless, there is a cost that is incurred for being in the new state. A cost of consequence $C \in$ COST OF CONSEQUENCES is a real number from a finite, discrete set of real number ≥ 0 that represent this cost. It is unitless.

Association between any two of the above elements are specified by mathematical relations. The following relations are defined in the trust based access control model.

i. DTA $\subseteq$ DEVICES $\times$ TRUST INDICATORS define the device – trust indicator assignment. It is a many to many relation where a device can have many trust indicators and a trust indicator can be associated with many devices.
ii. DTH $\subseteq$ DEVICES $\times$ TRUST_ HISTORY defines the device – trust history mapping. It is a one-to-one relation;

one device has one trust history and vice versa.

iii. TTT ⊆ TRUST TRIGGERS × TRUST_ INDICATORS specify the trust trigger – trust indicator assignment. It is a many to many relation where a trust trigger can cause change in value for many trust indicators and the same trust indicator can have a change in value caused by many trust triggers.

iv. DVT ⊆ DEVICES × TRUST VALUES define the device – trust value mapping. It is a many to one mapping from DEVICES to TRUST VALUES. A device can be associated with one and only one trust value. However, many devices can have the same trust value.

v. TAT ⊆ TRUST VALUES × ACCESS TYPES define the trust value needed for a specific access type. It is a many to many mapping.

vi. ATA ⊆ ACCESS TYPE × ACCESS define the access type to access mapping. It is a many to one mapping from ACCESS TYPE to ACCESS. Several access types may imply the same access; however, every access is of one specific type.

vii. ACP ⊆ ACCESS × PERMISSIONS defines the permissions needed for a specific access. It is a many to many relation.

viii. PCE ⊆ PERMISSIONS × COST OF EFFECTUATING _PERMISSION is a mapping that define the relation between a permission and the cost associated with implementing the permission. It is a many to many relations. A cost of a permission $c \in$ PCE is a real number from a finite set of numbers ≥ 0 that gives the unitless cost.

ix. PEC ⊆ PERMISSIONS × CONSEQUENCES is a mapping that define the many to many relation between PERMISSIONS and CONSEQUENCES. A permission consequence $\xi \in$ PEC gives the consequence of effectuating the permission.

x. CCS ⊆ CONSEQUENCES × COST OF CONSEQUENCES is a many to many mapping between CONSEQUENCES and COST OF CONSEQUENCES. An element $C \in$ CCS is a real number ≥ 0 that gives the specific cost associated with a specific consequence.

xi. Constraint $\min_i (\alpha.c(A_i) + \beta.C(A_i))$ defines an expression involving elements of PCE and CCS that needs to be minimized to select an appropriate set of access in a session.

VII. Conclusion

In this work, we take the position that over-reliance on cryptographic techniques in IoT is flawed because of various computational issues, scalability and management issues and implementation issues with existing public key based cryptographic techniques. We present a vision for an IoT ecosystem that achieves considerable practical security without using cryptography. It is based on creating unforgeable identities of IoT devices through behavioral fingerprinting that is strongly tied to the device, and on a trust framework for IoT devices that is not reputation based (unlike other trust models) but uses measurable and quantifiable properties of IoT devices to establish trust levels for them. We outline the architecture of the trust framework as well an access control model that uses the underlying trust model.

References

[1] "State of the IoT 2018: Number of IoT devices now at 7B – Market accelerating," https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/.

[2] B. Krebs, "Mirai IoT Botnet Co-Authors Plead Guilty-Krebs on Security," https://krebsonsecurity.com/2017/12/mirai-iot-botnet-co-authors-plead-guilty/, 2017.

[3] OCF, "OCF Security Specification Version 2," Open Connectivity Foundation, Technical Specification, 2018.

[4] R. A. Grimes, "4 fatal problems with PKI," http://www.infoworld.com/article/2942072/security/4-fatal-problems-with-pki.html, Jun. 2015.

[5] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki, "The DET curve in assessment of detection task performance," National Inst. of Standards and Technology, Gaithersburg MD, Tech. Rep., 1997.

[6] T. Kohno, A. Broido, and K. C. Claffy, "Remote physical device fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 2, pp. 93–108, 2005.

[7] R. Lippmann, D. Fried, K. Piwowarski, and W. Streilein, "Passive operating system identification from TCP/IP packet headers," in *Proc. of the Workshop on Data Mining for Computer Security*, vol. 40, 2003.

[8] J. Franklin, D. McCoy, P. Tabriz, V. Neagoe, J. V. Randwyk, and D. Sicker, "Passive Data Link Layer 802.11 Wireless Device Driver Fingerprinting." in *Proc. of the USENIX Security Symposium*, vol. 3, 2006, pp. 16–89.

[9] J. Pang, B. Greenstein, R. Gummadi, S. Seshan, and D. Wetherall, "802.11 user fingerprinting," in *Proc. of the 13th Annual ACM International Conference on Mobile Computing and Networking*, 2007.

[10] J. François, H. Abdelnur, O. Festor *et al.*, "Automated behavioral fingerprinting," in *Proc. of the International Workshop on Recent Advances in Intrusion Detection*. Springer, 2009, pp. 182–201.

[11] C. Arackaparambil, S. Bratus, A. Shubina, and D. Kotz, "On the reliability of wireless fingerprinting using clock skews," in *Proc. of the Third ACM Conference on Wireless Network Security*, 2010.

[12] A. Kurtz, H. Gascon, T. Becker, K. Rieck, and F. Freiling, "Fingerprinting mobile devices using personalized configurations," *Proc. of Privacy Enhancing Technologies*, vol. 2016, no. 1, pp. 4–19, 2016.

[13] V. Brik, S. Banerjee, M. Gruteser, and S. Oh, "Wireless device identification with radiometric signatures," in *Proc. of the 14th ACM International Conference on Mobile Computing and Networking*. ACM, 2008, pp. 116–127.

[14] S. Jana and S. K. Kasera, "On fast and accurate detection of unauthorized wireless access points using clock skews," *IEEE Transactions on Mobile Computing*, vol. 9, no. 3, pp. 449–462, 2009.

[15] S. V. Radhakrishnan, A. S. Uluagac, and R. Beyah, "GTID: A technique for physical deviceanddevice type fingerprinting," *IEEE Transactions on Dependable and Secure Computing*, vol. 12, no. 5, pp. 519–532, 2014.

[16] D. Formby, P. Srinivasan, A. Leonard, J. Rogers, and R. A. Beyah, "Who's in Control of Your Control System? Device Fingerprinting for Cyber-Physical Systems." in *Proc. of the Network and Distributed Systems Security Symposium*, 2016.

[17] T. Van Goethem, W. Scheepers, D. Preuveneers, and W. Joosen, "Accelerometer-based device fingerprinting for multi-factor mobile authentication," in *Proc. of the International Symposium on Engineering Secure Software and Systems*. Springer, 2016, pp. 106–121.

[18] M. Miettinen, S. Marchal, I. Hafeez, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "IoT Sentinel: Automated device-type identification for security enforcement in IoT," in *Proc. of the 37th IEEE International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2017, pp. 2177–2184.

[19] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, "Behavioral fingerprinting of iot devices," in *Proc. of the 2018 Workshop on Attacks and Solutions in Hardware Security*, Toronto, Canada, 2018.

[20] S. Marchal, M. Miettinen, T. D. Nguyen, A.-R. Sadeghi, and N. Asokan, "AuDI: Toward Autonomous IoT Device-Type Identification Using Periodic Communication," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1402–1412, 2019.

[21] A. Jøsang, *Subjective Logic*, ser. Artificial Intelligence: Foundations, Theory, and Algorithms. Springer International Publishing, 2016.