Efficient and Privacy-Preserving Access to Sensor Data for Internet of Things (IoT) based Services

Paramasiven Appavoo, Mun Choon Chan, Anand Bhojan, Ee-Chien Chang
National University of Singapore
Email: {pappavoo,chanmc,banand,changec}@comp.nus.edu.sg

***Abstract*—As a major driver of the Internet of Things (IoT), sensors are harvesting data, from their environments, that service providers make use to trigger the appropriate services. These service providers require access to a wide range of personal data, which are often sensitive. In this paper, we propose a lightweight privacy-preserving trust model based on the observation that a large class of applications can be provisioned based on simple threshold detection. The key issue we address in this work is how to minimize privacy loss in the presence of untrusted service providers so that providers are prevented from disclosing information to third parties for secondary uses. Our work can be considered as a lightweight approach to functional encryption (FE) for privacy-preservation. The main algorithm in the proposed model is a uniformization scheme that uses a combination of sensor aliases to hide the identity of the sensing source and per-function initialization vector to reveal information only to relevant service providers. We have implemented a prototype of the proposed scheme on TelsoB, thereby demonstrating the feasibility of the proposed scheme on resource-constrained devices.**

I. Introduction

IoT is the next big wave that is expected to transform the way people and objects interact. IoT devices include sensors embedded in the environment and wearables such as biometric sensors that can closely monitor one's health condition and life style.

One of the challenges which must be overcome before widespread deployment of services that relies on sensitive data collected by IoT devices is privacy ([1], [19], [11], [20], [6], [22]). Due to closeness of the sensors to the physical realm, privacy issues must be tackled by design to enable wide acceptance by users [19].

Earlier work on sensor data aggregation assumed the use of trusted aggregator and thus cannot protect users against untrusted aggregator. More recent work ([12], [17]) consider data aggregation using untrusted aggregator. While these schemes provide better data privacy protection, they are designed to perform some kind of data aggregation operations (e.g. summation, average or minimum). The computation overhead can be significant and impose substantial burden on resource constrained devices that are common in IoT environments.

The motivation in our work is based on the observation that a large class of applications can be provisioned based on simple detection of whether sensor values have crossed some given thresholds. For example, consider the case of a wellness/health monitor service that will automatically summon a visit by health care workers if the blood pressure and heart rate of a user has crossed specific thresholds. In fact, any form of application scenarios that involve anomaly detection can be supported by such a framework.

Our model consists of three components: the sensors, (unstrusted) data store and service providers. The sensors are pre-configured to generate triggers based on sensor values crossing given thresholds. Sensors transmit information on whether triggers are activated or not to an untrusted data store. The data store only provides storage facility and performs no computation. Finally, a user subscribes to various services by telling the respective service providers where to get the data and how to extract the trigger information.

Our proposal has the following advantages. First, processing on sensor is minimum and can be easily implemented on resource constrained device. Second, since the data store and service provider are separate entities, service provisioning is now open to external third party providers and a user have the flexibility to dynamically configure the service he/she wants.

Our privacy-preserving design can be considered as a simple form of functional encryption [5]. The data published by sensors only relates to the trigger information. Both the raw sensor readings and the trigger conditions are stored on the sensors and are not known to the data store. The trigger information is stored in an encrypted form and a service provider is provided with the keys to extract only the information relevant to the service to be provided. In addition, through the use of aliases, the data store cannot identify the sensor whom it receives the data from and through a uniformization process, an observer cannot deduce whatever a trigger has been activated or not. To validate our claim on efficiency, we have implemented a prototype of the proposed system on a TelosB mote running Contiki. Our evaluation shows that our design indeed requires little resources and incurs minimum overhead.

The rest of this paper is organized as follow. Section II presents related work and Section III presents a sample application scenario to motivate our work. We present our model in Section IV and details of the algorithms implemented in Section V. Section VI presents evaluations of our prototype implementation and we conclude in Section VII.

II. Related work

Privacy-preserving methods have been extensively used in data publishing, data mining, location-based services, data aggregation in both mobile sensing and wireless sensor networks, and other areas. Recently, with privacy concern at the heart of IoT, research outcomes on privacy, with respect to same, started emerging.

The traditional techniques employed in data publishing and also extended to other domains are mainly k-anonymity[23], l-diversity [18], t-closeness [16], and differential privacy [7]. These techniques are designed to support data mining and solved a problem that is different from the stated problem in this work.

The privacy concern with location-based services arises even before the use of such services as pointed out in [14]. The computation of the location information is preferably performed on the mobile device using GPS instead of a remote server computing same for a device based on beacons sensed. An example of the third party computing the location was implemented in [15]. In order to reduce privacy invasion from the use of location-based services, the use of different pseudonyms was suggested in [2]. Moreover, the well-known k-anonymity and an extended version, known as historical k-anonymity, in [10] and [3] respectively, were applied in the context of location-based services to obfuscate the location reported by one individual. Apart from the pseudonym concept that can be applied to hide the identity of the sensor, such techniques cannot be utilized when a user application provides a personalized service, where only true positives are acceptable, by accessing the users health-related sensors.

There are work done on privacy preserving data aggregation in wireless sensor networks. PDA [12] proposes two schemes that tradeoff between computational resource and bandwidth resource. Li and Cao propose [17] a privacy-preserving data aggregation scheme based on homomorphic encryption for sensor data collection by an untrusted aggregator.

When it comes to IoT-specific privacy works, [8] proposed the use of data tagging. The scheme controls the flow of information based on the tag it received at creation time. While [22] proposed a model where access to sensitive data is restricted to authorized users and devices, [13] proposed an access control protocol that allows IoT users to share data at different privacy levels. Part of their system also included the generalization of sensor data to achieve k-anonymity.

All the above privacy-related works, whether related to IoT or not, do not prevent an application or service provider from disclosing a user's sensitive information. A recent effort in changing how encryption can be varied to be more flexible is the idea of functional encryption (FE) [4], [5]. An example of FE technique is Attribute-Based Encryption (ABE), which include Ciphertext-Policy (CP) and Key-Policy (KP). However, these ABE systems have high overhead [5] and are not suitable for implementation on resource constrained IoT devices.

III. Application scenario - IoT wellness server

While the proposed scheme is not so attractive to analytics applications requiring raw sensor data, it is expected to work pretty well for threshold-based applications. An example of a system using the proposed scheme is shown in Figure 1. This example is a personalized wellness server for patient monitoring. The patient/user starts by registering on the wellness IoT server. He provides his (encrypted) personal details and a preferred list of doctors/hospitals. He then configures his sensors to send the trigger to the wellness server when the trigger conditions are met.

In this scenario, there are two services subscribed: (1) issuing an emergency call, and (2) sending patient data to user's preferred doctor for recommendations. While these two services may be triggered by the same set of sensors, they may have different trigger conditions. For example, when the heart rate and blood pressure readings have been on an elevated level for some time, a doctor may be consulted. On the other hand, if these values are dangerously high, an ambulance can be called immediately. It is easy to imagine that many other services can be derived from additional sensors and different sensor combinations.

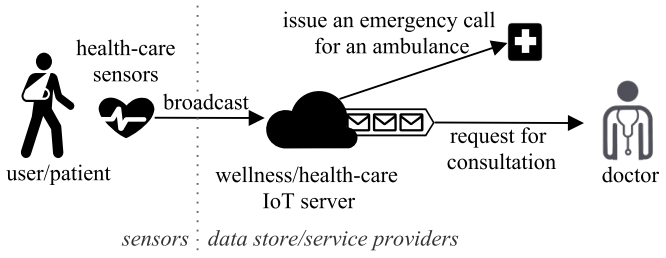

Fig. 1. Scenario with a threshold-based application.

With the proposed scheme, the user has control over what is being shared with whom. Many applications can be subscribed to without having the sensitive information being stored on the data store or unnecessarily revealed to the application providers.

IV. Proposed privacy-preserving model

The proposed model is designed to meet the following objectives. First, the IoT servers or service providers should be prevented from disclosing sensed values, types of sensors, and user preferences, in the form of triggering conditions or functions, to third-parties (e.g. health insurance companies or potential employees) for secondary uses. Second, by observing the communications between the sensors, data store and service providers, it should not be possible to determine the source of the data as well as whether any particular trigger condition has been met. Third, even though a user subscribes to multiple services, each service provider should get access only to relevant information needed for the service.

In this work, we consider two kinds of adversaries

- A *semi-trusted service provider* that has access to the outcome of a selected set of trigger conditions. Based on these outcomes, it determines the course of action (e.g. call the ambulance) even though it does not know the user configurations/conditions/functions (e.g. Blood pressure (Systolic) > 180). Note that the provider is semi-trusted because it is given access to only a user-defined subset of all possible sensors plus trigger conditions combinations.
- A *passive listener* that can observe the communications between the sensor and data store as well as data store and application provider.

We will mainly consider the case of a semi-trusted application provider. The key question is as follows. Given that the application provider needs to know the outcome of the sensor

trigger condition, how can the privacy loss be minimized while the service is being provided? We address this question in two steps. First, we look at the problem in a more general manner using mutual information. Next, we consider the problem in a more detailed level in terms of minimizing privacy loss. The case of a passive listener can be handled through standard symmetric key encryption.

A. Privacy loss in terms of mutual information

Privacy loss can be considered in a more general manner using mutual information; a similar approach to work [21].

First, let us define the attributes of interest:

- S, the random variable for the set of sensors that can be utilized;
- V, the random variable for the set of sensed values (assuming we are interested in these sensor values at a particular time t);
- T, the random variable for the set of (binary) outcomes for the trigger conditions.

We then have (1) H, the entropy function, that measures uncertainty with respect to the random variables, and (2) I, the mutual information function, that returns how much information about unknown variables is revealed when the outcome of a certain variable is known.

In our model, privacy loss is related to what information can be deduced based on known information. The uncertainty about the connected sensors and their respective actual readings, $H(S,V)$, is affected by how much it is conditioned by T. Therefore, $H(T)$ also includes mutual information $I((S,V);T)$, where $I(S,V;T) = H(S,V) - H(S,V|T)$. $I(S,V;T)$, the privacy loss (shaded region shown in figure 2), represents the amount information being disclosed about the connected sensor and its respective reading.

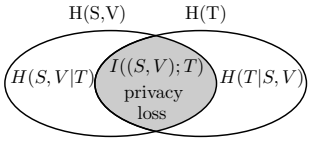

Fig. 2. Privacy loss is about how much is learned about the sensor and its respective value, given that there was a trigger for a service.

The general form of the distribution about (1) the set of sensors used (S), (2) the set of sensor values (V), and (3) the potential of a particular value from a particular sensor causing a trigger (T) is useful to understand the potential privacy loss from general inferences. For this study, the distributions considered are uniform (U), power law (P) and normal (N). Power law distribution is useful to show cases where only a few instances of combinations of the considered element are much more likely to affect the outcome. It might be the case that only a few values, over a few attached sensors, have much higher probabilities of causing a trigger while the majority of values from most sensors have a very low probability of causing a trigger.

The results with respect to a number of sensors with a set of possible values, as an average over 1000 runs, are shown in figure 3. The parameters of the distributions used in the simulation study are as follows: (1) uniform: probability of $\frac{1}{16}$ for the 16 sensors and 16 discrete values considered and $\frac{1}{2}$ for the trigger, (2) power law: alpha was set to 0.4, and (3) standard normal with $\mu = 0$ and $\sigma = 1$.

$H(S,V)$ is the maximum information that can be inferred about sensors used and their values. The maximum privacy loss for all the respective combinations (distributions) is shown in figure 3 with the dotted red line. This pattern is almost the same irrespective of the amount of sensors and their possible readable values. Privacy loss is bounded by $H(T)$. Given that T can have only two possible outcomes, the maximum of $H(T)$ is 1. Therefore, privacy loss with the proposed model is less or equal to 1 bit. Moreover, it is to be noted that privacy loss is minimized when the $T \sim U$.

B. Privacy Loss with Semi-Trusted Application Provider

In order to provide services (e.g. an ambulance is called), some of the outcome of the triggers setup by a user needs to be revealed to the relevant service provider. At the same time, it is also important that any additional information that are not needed to provide the service should not be known to the provider. Hence, we called the provider semi-trusted.

As discussed previously, in order to minimize loss of privacy, T should ideally be uniformly distributed. However, T is obviously not uniformly distributed in general. In order to minimize privacy loss, we introduce a uniformization scheme to transform the outcomes available to the semi-trusted service provider so that the distribution of T observed is uniformly distribution independent of the actual distribution.

This high level idea of the proposed trigger uniformization scheme is shown in figure 4. As input, the scheme requires the list of physical sensors used, as well as the probabilities of meeting various triggering conditions for each physical sensors. A physical sensor is then mapped to many (sensor) aliases.

The objective of the scheme is that by observing output, one cannot determine the outcome of the triggers and their actual respective origins. Only relevant service providers are granted the keys to recover the set of trigger outcomes that are needed to implement the requested services. Still, the actual sensor, where the trigger originated, remains not only hidden but also made hard to guess. Conceptually, this is achieved by masking a real sensor with sensor aliases that have the same probability of triggering the service. Details of the implementation are given in the next section.

V. Privacy-preserving sensor data dissemination

The basis for the proposed algorithms is the trust and uniformization models described in section IV. The proposed algorithm can be considered as a lightweight model of FE, that not only hides the plaintext but also obfuscate the potential sources of the trigger.

A. Overview

Table I and figure 5 highlights the required functions and the relationships between them respectively. The functions

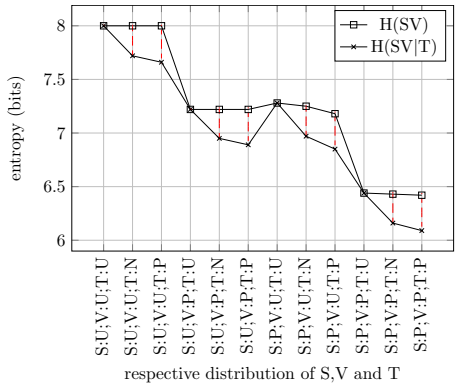

Fig. 3. Privacy loss, red dashed lines, with respect to the distribution of S,V and T (case of 16 sensors and 16 values in sensor range)

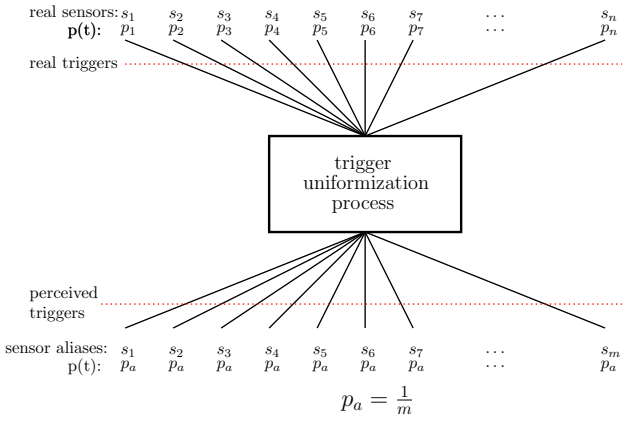

Fig. 4. Trigger uniformization model: While real sensors have different probabilities of triggering an event, this tendency is uniformized with sensor aliases.

'uniformize and encrypt' and 'decrypt' are executed periodically as and when data needs to be published and interpreted respectively. The rest of the functions are executed once.

TABLE I. SUMMARY OF FUNCTIONS

Functions	Descriptions
(1) Setup: estimate probability	Estimate the probability that a given function will be TRUE, based on sample data collected
(2) Keygen: generate aliases/functions	Generate the aliases with set of functions and keys for uniformization and encryption respectively
(3) Uniformize and encrypt	Maximize the entropy of finding out the real sensor behind the trigger of an event and encrypt data to be disseminated
(4) Decrypt	Decrypt the outcome(s) of the function(s) applied on the data

The functions are briefly described as follows:

1) Setup: The 'setup' process takes as inputs the distribution of the sensor data and a list of functions. A function could be a simple condition, like '≥ 30', that is evaluated against sensor data. The outcome from the 'setup' is the estimated probabilities of the respective functions being evaluated to TRUE.

2) Keygen: The 'keygen' process then takes the output from the 'setup', the list of functions and the number of maximum aliases possible. It generates a set aliases with respective set of functions. The number of appearances of a particular function over the aliases depends on the probability that the particular function is evaluated to TRUE. The number of appearances is useful in the masking process, that we refer to as 'trigger uniformization', of a particular function over a particular physical sensor. Finally, the keys to identify a particular {sensor alias,function} pair is shared with semi-trusted parties and is used to encode and decode the disseminated data.

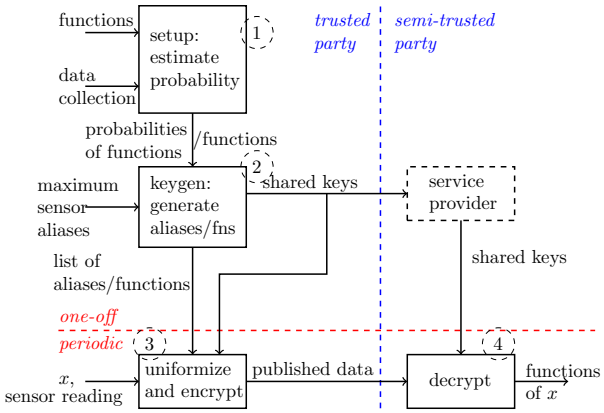

Fig. 5. Relationships between functions of the proposed scheme.

3) Uniformize and encrypt: The 'uniformize and encrypt' process first evaluates the functions with respect to the current sensor reading. Next, based on the outcome, it selects the sensor aliase(s) over which the outcome is broadcasted. This is done in such a way that an observer sees a trigger, from a function evaluating to TRUE, to be equally likely coming from multiple sensors. Finally, the data is encrypted before dissemination.

4) Decrypt: The 'decrypt' process allows a semi-trusted service provider, with the appropriate keys, to decide whether a particular service is being requested. Also, it is to be noted that all keys, with respect to the desirable triggers, are shared 'offline' with the respective service providers.

B. Setup and key generator

The main functions of this subsection are defined as follows:

$$\begin{aligned} P &\leftarrow setup(F) \\ \forall i, \exists j : sk_{ij} &\leftarrow keygen(f_i, p(f_i = 1)) \\ f_i &= \{r_i, op_i\}, op_i \in \{<, >, \approx\} \\ sk_{ij} &= \{a_j, cx_{ij}, ivx_{ij}, AES\_sk_{ij}\} \\ AES\_sk_{ij} &\in \{key_{ij}, \emptyset\} \end{aligned}$$

sk_{ij} is mainly a set composed of an optional standard symmetric encryption key and key indexes required to hide/recover the outcome of f_i over a sensor alias j. The key indexes are pointers to specific bits of the broadcast space. An instance of sk can be "$(10.0.0.1, 65, 50, \emptyset)$". More details about the indexes are found in the subsection V-C. At a higher cost, in terms of both space and computation, it is also possible to use standard symmetric encryption algorithm to protect the outcome of the functions. Then, $AES\_sk_{ij} \neq \emptyset$.

λ	: secret parameters for defining the functions
a	: sensor alias (reference to a physical sensor, e.g IP address)
cx	: index of encrypted outcome from the function
f	: function/condition
i	: represents the i^{th} function
ivx	: index of initialization vector
j	: represents the j^{th} sensor's alias
op	: operator for the condition/function
p	: the probability function
r	: reference value
sk	: shared key
AES_sk	: shared key of AES or any other symmetric encryption algorithm
F	: set of functions
P	: set of probabilities for the respective functions to be TRUE

While F and the set of aliases, A, are shared with trusted parties only, selected sk_{ij} can be shared with semi-trusted parties as well. In the setup phase, the probabilities can either be estimated through measurements or be given by the developer. The pseudocode for 'keygen' is given below. The estimated probabilities from the setup phase are used in 'keygen' to compute the number of required aliases that f_i has to appear in order to allow for the trigger uniformization.

algorithm: keygen(F, P)

```
repeat
    num_aliases[i] = ceil(p(f_i = 1) * MAX_ALIASES)
    increment i
until all functions/conditions in F are done
repeat
    alias[j] = RANDOM_ALIAS()
    increment j
until MAX(num_aliases[])
reset j
repeat
    reset i
    repeat
      if (num_aliases[i] < j)
        cx_ij = RANDOM_FREE_SLOT()
        ivx_ij = RANDOM_FREE_SLOTS()
        sk[i][j] = (alias[j], cx_ij, ivx_ij)
      increment i
    until all f_i for alias[j] are done
    increment j
until MAX(num_aliases[]) are done
```

With reference to the 'keygen' algorithm, it is to be noted that the constant MAX_ALIASES, maximum number of possible sensor aliases, depends on the resources available on the sensor platform. Section VI shows the possible values when our prototype was implemented on a particular platform. As the c and iv are randomly placed in the output, the keys sk contains information on where to retrieve the required outcome.

C. Uniformization and encryption

The encoding process includes a uniformization process and a standard encryption function and is accomplished as shown in figure 6. A number of functions is applied on the sensor reading. The respective outcomes are encrypted and published. In the process, based on the outcomes, an alias j (or multiple aliases) is selected to potentially uniformize the source of the trigger. The process for generating the broadcast space, B, can be summarized as::

$$\forall i, \exists_{=1} j \iff f_i(x) = 1 : \\ B_j \leftarrow encrypt(f_i(x) \rightarrow \{0,1\}, sk_{ij})$$

The following is achieved:

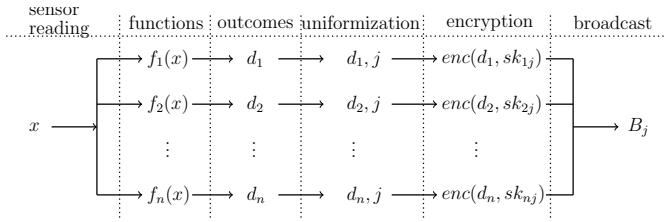

Fig. 6. Encoding process: from sensor reading to published data

1) Broadcast uniformization: As shown in figure 7, the slot at the index cx_{ij}, of the broadcast space B for the sensor alias j, holds the value c_{ij}, where $c_{ij} = f_i(x) \oplus \ldots$ (the XORed outcome of iv_{ij} individual bits). The number of bits in the iv must be at least 2 so as to ensure that c_{ij} is equally likely to appear as 0 or 1, whether $f_i(x)$ is either 0 or 1. As such, all bits across successive broadcasts are equally likely to be 0 or 1, irrespective of (1) x, and (2) f_i. As the entropy of a bit is at its highest, the uncertainty for potential eavesdroppers is maximized.

Each f_i is associated with a distinct iv so as to enable a flexible distribution of the data. This approach allows a service provider to observe only what he is granted access, as far as possible.

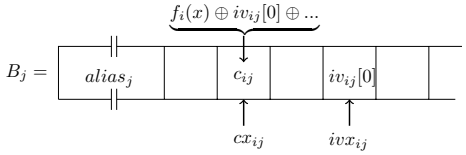

Fig. 7. Broadcast structure: B_j; indexes cx_{ij} and ivx_{ij} are randomly assigned at 'keygen' phase. $alias_j$ or a_j is used to identify a sensor alias. With 4 bytes, it is possible to have 2^{32} possible aliases. The source address of existing protocols' header field can be exploited, whenever possible. In another case, for example using REST, URLs with same domain but different paths could correspond to different aliases.

2) Trigger uniformization - General case: When $f_i(x)$ evaluates to 1, this outcome appears on only one alias, say a, for the duration that it stays high. The next time the evaluation $f_i(x)$ shifts from 0 to 1, it is again maintained on one only alias, say b where $b \neq a$. This is so unless the number of aliases for f_i is 1. As such, if we assume that the data distributions captured across multiple sensors are reliable and they can send data to a service provider, the above approach maximizes the entropy of inferring the sensor behind source of the trigger. This is so as the number of aliases, on which f_i can appear, is proportional to the probability that f_i evaluates to 1. We consider two cases of trigger uniformization, starting from the most expensive, but most effective, down to a less costly one.

3) Trigger uniformization: Case 1 - one sensor alias supporting one function only: In this scenario, as shown in figure 8, one sensor alias is related to only one function while the latter is related to multiple sensor aliases. The number of sensor aliases to which f_i is related is based on the $p(f_i = 1)*$ maximum possible sensor aliases. This approach allows for an application, requiring multiple evaluated functions, where more than one can be TRUE at anytime, from a 'physical' sensor. While the potential physical sources of the triggers

are hidden, this technique is quite expensive when used in a sensor environment that uses wireless broadcast. The algorithm is given below.

algorithm (case 1): uniformize and encrypt(x, F, SK)

for all functions in F
  if evaluate$(f[i], x) == 1$:
    j = select_alias_with_i_randomly(i)
    iv_{ij} = rand()
    $B_j[sk[i][j].cx] = (\text{evaluate}(f[i], x) = 1) \oplus iv_{ij}[0] \oplus iv_{ij}[1] \oplus \ldots$
    $B_j[sk[i][j].ivx] = iv_{ij}$
    increment i
    broadcast B_j

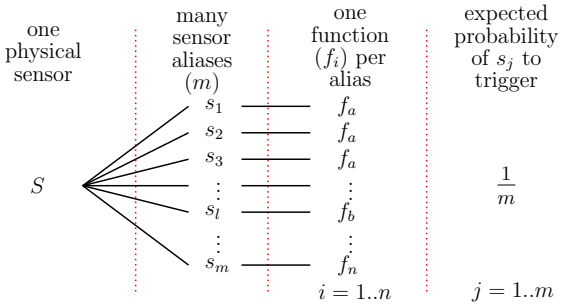

Fig. 8. Ideal trigger uniformization process: each time a function, say f_x, holds true, it potentially appears on a 'different' sensor, using aliasing. The number of occurrences of f_x depends on its probability to be TRUE. As such, the expected probability for s_x to be the source of the trigger is uniformized to $\frac{1}{m}$, thereby maximizing entropy.

4) Trigger uniformization: Case 2 - one sensor alias supporting multiple functions: In this case, one broadcast disseminate the outcome of several functions. If the functions check for the sensor reading in mutually exclusive range, then the algorithm in case 1 is slightly modified (as shown below; but can be further optimized) to fulfill this requirement; otherwise, a set cover approximation solution can be applied to minimize the number of broadcast. While it may be a slightly less effective uniformization process, one broadcast allows service providers of non-triggered events to know that the sensor is alive. This is the scheme implemented in Section VI.

algorithm (case 2): uniformize and encrypt(x, F, SK)

for all functions in F
  if evaluate$(f[i], x) == 1$:
    j = select_random_alias_supporting_i(i)
    break
for all functions in F
  iv_{ij} = rand()
  $B_j[sk[i][j].cx]$ =evaluate$(f[i], x) \oplus iv_{ij}[0] \oplus iv_{ij}[1] \oplus \ldots$
  $B_j[sk[i][j].ivx] = iv_{ij}$
  increment i
broadcast B_j

Using the above algorithm, by selecting the proper alias based on x, the sensor reading, trigger uniformization is possible. Moreover, one instance of $f_i(x)$ requires at least 3 bits, one for the encrypted $f_i(x)$ bit outcome and at least two bits for the iv. Other encryption methods can also be applied to hide the outcome of $f_i(x)$. For example, if AES scheme is used instead, the required update is as shown:

$$B_j[sk[i][j].cx] = \text{enc\_AES}(\text{evaluate}(f[i], x),\ AES\_sk[i][j], iv_{ij})$$

As the combinations of functions required by service providers are unknown in advance, the encryption/decryption of each specific function is associated with one particular AES key, say AES_sk_{ij}. Thus, the use of AES can be highly expensive as one instance of $f_i(x)$ requires 128 bits.

D. Decryption

Given that f_i matches a service provider's requirement, the decryption algorithm, $decrypt$, shown below provides level of desired utility while (1) preserving the sensor reading, x, and (2) hiding the real trigger's source behind numerous aliases.

$$\forall i, j : f_i(x) \leftarrow decrypt(B_j, sk_{ij})$$
$$x_i \leftarrow g(f_i(x), f_i)$$
$$h : a_j \rightarrow sensor_m$$

g, h : utility functions
A : set of aliases
M : set of sensors' real identity

Trusted parties, for example a family doctor, have F with which a function g allows them to get x or a reference to x, say x_i, as the related function can be a simple condition expressed over x. They also have a function h and A to enable the mapping from aliases to actual sensors' identity/type in M.

algorithm: decrypt(B_j, sk_{ij})

for all functions referenced by $sk[*][j]$
  $c_{ij} = B_j[sk[i][j].cx]$
  $iv = B_j[sk[i][j].ivx]$
  $f[i] = c_{ij} \oplus iv_{ij}[0] \oplus \ldots$

E. Disseminated data with privacy-preserving property

The disseminated data exhibits the desired privacy-preserving objective with the following properties:

1) Resistance to eavesdropping attacks: The scheme is resistant to eavesdroppers as the entropy is maximized with each bit of the broadcast being equally likely to be 0 or 1 irrespective of the functions outcome, sensors' type or reading.

2) Resistance to collusion attacks: Each colluding attacker knows only the outcome of an unknown set of functions over a set of unidentified sensors. The maximum information gained remains at most one bit with respect to the sensor's identity/type and its respective reading. This is so as $H(S, V) - H(S, V|T) \leq 1$.

3) Resistance to inference attacks: Inference attacks are hard as the the probability that the trigger is caused by all possible sensors, irrespective of their values, is made equally likely with the trigger uniformization approach. Moreover, the entropy is expected to increase significantly as the number of sensor aliases can significantly outnumber the physical sensors.

VI. Implementation and Evaluation on a Resource-Limited Platform

A. Setup

The proposed scheme has been implemented on a Telos Rev B, with an MSP430 microcontroller of 8 MHz with a

programmable flash memory of 48 KBytes and RAM of 10 KBytes. Rime broadcast was used for message exchanges between two motes, where one basically implemented the 'setup', 'keygen' and 'uniformize and encrypt' functions while the other one implemented the 'decrypt' function.

In this experiment, the 'setup' generates the list of functions, where each one is made up of two simple conditions that defines a range for the sensor value. All data of the payload were manipulated in the binary form and the size of the iv is two random bits to encrypt each respective outcome. The power measurements were done with a power monitor operating at a frequency of 5 kHz.

B. Time and energy requirements

1) One-off functions: Table II shows the time taken and energy consumed to execute the setup & keygen functions. These are one-off operations. The energy consumption of a broadcast, using rime in Contiki, a periodic function to broadcast the payload of the specified size, is shown as a reference to have an idea about the energy requirements of the proposed functions, in table II and figure 9.

2) Recurring functions: The energy consumption for encryption and decryption is shown in table III and figure 10. As defined for the possible alternative to complete the encryption process in section V, part of the encryption process was swapped with AES, using the hardware implementation on TelosB, to show the difference in energy consumption. The current drawn for most processing was 2.19 mA while that including the hardware-level AES encryption method is 2.54 mA.

C. Space requirements

Table II shows the maximum possible functions and aliases with respect to the payload size and memory size respectively using the TelosB mote. For most combinations, the maximum RAM usage of 10 KB was reached while the ROM usage was about 28 KB, including the OS functions. The latter alone uses 5 KB and 21 KB of RAM and ROM respectively. It is to be noted that the maximum number of aliases, shown, is not applicable when substituting for AES sub-function. Moreover, when AES is used, the maximum number of functions in a 512 bits payload would be only 4 as shown in figure 9. This is so as one key is associated with an instance of $f_i(x)$ to enable a flexible distribution of keys with respect to the desired set of functions.

TABLE II. MAXIMUM POSSIBLE FUNCTIONS AND ALIASES WITH ENERGY CONSUMPTION FOR ONE-OFF OPERATIONS

payload size (bits)	max functions as per payload size	max aliases with all functions	frequency of execution		
			one-off		periodic
			total (setup & keygen)		rime broadcast as a reference
			sec	mJ	mJ
8	2	200	0.24	1.6	5.1
16	5	87	0.32	2.08	5.07
32	10	43	0.39	2.57	5.17
64	21	20	0.57	3.72	5.34
128	42	10	0.90	5.93	5.64
256	85	5	1.59	10.43	6.07
512	170	2	2.67	17.54	6.62

TABLE III. ENERGY CONSUMPTION - ENCRYPTION & DECRYPTION

number of functions	algorithms					
	uniformization and encryption		uniformization and encryption with AES sub-function		decryption	
	ms	μJ	ms	μJ	ms	μJ
1	0.97	6.37	2.12	16.15	0.41	2.69
2	0.98	6.44	2.13	16.23	0.79	5.19
5	1.58	10.38	3.39	25.83	1.12	7.36
10	2.37	15.57	6.23	47.47	1.43	9.4
21	4.17	27.40	12.63	96.24	2.58	16.95
42	7.96	52.30	24.92	189.89	5.11	33.57
85	15.25	100.19	50.09	381.69	10.34	67.93
170	30.1	197.76	99.58	758.80	20.61	135.41

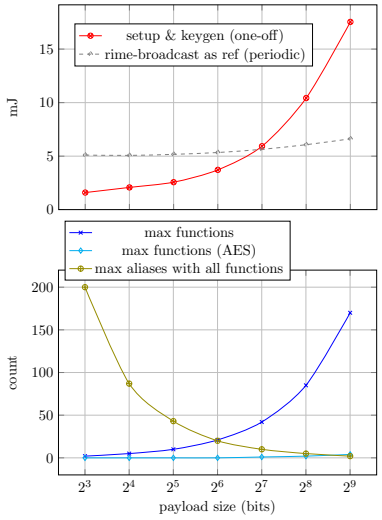

Fig. 9. Energy consumed (mJ) at max possible functions and aliases with respect to payload size

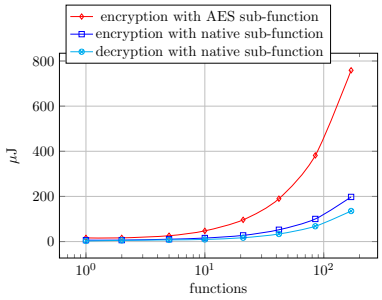

Fig. 10. Energy consumed (μJ) for encrypting/decrypting all functions' respective outcomes.

D. Expected performance of existing FE techniques on resource constrained devices

In this section, we evaluate the performance of existing FE techniques like KP-ABE and CP-ABE. The outcome of each function f_i is encrypted under a particular attribute for a flexible distribution, given that the requirements of service providers is not always known in advance. In order to perform this comparison, we ported our code to a desktop version. The FE code available at [9] was used. The experiments were performed on a virtual machine running Ubuntu 64-bit with 4 CPU cores of 3.4 GHz each and 4 GB of RAM. Moreover, in order to to estimate the time taken when using each of

the schemes, the value of the 'user' variable from the 'time' command in Linux was used.

The results shown in table IV are with respect to 170 attributes (mapped to case of 170 f_i in our proposed scheme), one per f_i. The expected performance on TelosB was computed using linear extrapolation. It is clear that the existing FE techniques are not practical on low-end sensor devices, even when the evaluation of f_i was excluded from the existing FE techniques. Similarly, the lightweight ABE scheme proposed

TABLE IV. PERFORMANCE COMPARISONS WITH EXISTING FE TECHNIQUES

CP-ABE		KP-ABE		proposed scheme: native	
setup & keygen	encryption (excluding evaluation of f_i)	setup & keygen	encryption (excluding evaluation of f_i)	setup & keygen	uniformization and encryption (including evaluation of f_i)
time taken, with Ubuntu 64 on VM, in seconds					
28.95	27.08	25.31	24.53	0.000994	0.000262
time taken, with Contiki OS on TelosB, in seconds					
values as computed with linear extrapolation				*values as measured*	
104,380.21	97,618.25	91,238.21	88,451.73	2.67	0.03

by [24] encryption process is expected to take approximately 4000 seconds on a TelosB mote, while this amount is expected to be twice for decryption process. This is computed on the basis that their encryption process takes approximately $\frac{1}{66}$ times the operations of CP-ABE, that consisted of encryption, decryption and bilinear mapping.

VII. CONCLUSION

Using the proposed trust information model, service providers do not gain additional information about the end-user when the outcome of the triggered event is in the public domain. Inferences, that can be made, are based on data that are publicly available. This has also been demonstrated with low mutual information between a trigger and the connected sensors with their respective possible triggering values. On top of that, the concept of trigger uniformization increases the uncertainty with which a sensor can cause a trigger. Finally, unlike existing FE techniques, our proposed scheme involves no expensive operation. We have also shown through our implementation and evaluation that the proposed scheme can be implemented on resource constrained devices as it is energy efficient and has low memory requirement.

VIII. ACKNOWLEDGEMENT

This work was supported in part by the Agency for Science, Technology and Research (A*STAR), Singapore, under SERC Grant 1224104049.

REFERENCES

[1] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.

[2] Alastair R Beresford and Frank Stajano. Location privacy in pervasive computing. *IEEE Pervasive computing*, (1):46–55, 2003.

[3] Claudio Bettini, X Sean Wang, and Sushil Jajodia. Protecting privacy against location-based personal identification. In *Secure data management*, pages 185–199. Springer, 2005.

[4] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography*, pages 253–273. Springer, 2011.

[5] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: a new vision for public-key cryptography. *Communications of the ACM*, 55(11):56–64, 2012.

[6] Eleonora Borgia. The internet of things vision: Key features, applications and open issues. *Computer Communications*, 54:1–31, 2014.

[7] Cynthia Dwork. Differential privacy. In *Proceedings of the 33rd International Conference on Automata, Languages and Programming - Volume Part II*, ICALP'06, pages 1–12, Berlin, Heidelberg, 2006. Springer-Verlag.

[8] David Evans and David M Eyers. Efficient data tagging for managing privacy in the internet of things. In *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, pages 244–248. IEEE, 2012.

[9] Matthew Green and Joseph Ayo Akinyele. libfenc - the functional encryption library. https://code.google.com/p/libfenc/, 2010.

[10] Marco Gruteser and Dirk Grunwald. Anonymous usage of location-based services through spatial and temporal cloaking. In *Proceedings of the 1st international conference on Mobile systems, applications and services*, pages 31–42. ACM, 2003.

[11] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013.

[12] Wenbo He, Xue Liu, Hoang Nguyen, Klara Nahrstedt, and Tarek Abdelzaher. Pda: Privacy-preserving data aggregation in wireless sensor networks. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*, pages 2045–2053. IEEE, 2007.

[13] Xin Huang, Rong Fu, Bangdao Chen, Tingting Zhang, and AW Roscoe. User interactive internet of things privacy preserved access control. In *Internet Technology And Secured Transactions, 2012 International Conference for*, pages 597–602. IEEE, 2012.

[14] John Krumm. A survey of computational location privacy. *Personal and Ubiquitous Computing*, 13(6):391–399, 2009.

[15] Anthony LaMarca, Yatin Chawathe, Sunny Consolvo, Jeffrey Hightower, Ian Smith, James Scott, Timothy Sohn, James Howard, Jeff Hughes, Fred Potter, et al. Place lab: Device positioning using radio beacons in the wild. In *Pervasive computing*, pages 116–133. Springer, 2005.

[16] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 106–115. IEEE, 2007.

[17] Qinghua Li and Guohong Cao. Efficient and privacy-preserving data aggregation in mobile sensing. In *Network Protocols (ICNP), 2012 20th IEEE International Conference on*, pages 1–10. IEEE, 2012.

[18] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3, 2007.

[19] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497–1516, 2012.

[20] Rodrigo Roman, Jianying Zhou, and Javier Lopez. On the features and challenges of security and privacy in distributed internet of things. *Computer Networks*, 57(10):2266–2279, 2013.

[21] Lalitha Sankar, S Raj Rajagopalan, and H Vincent Poor. Utility-privacy tradeoffs in databases: An information-theoretic approach. *Information Forensics and Security, IEEE Transactions on*, 8(6):838–852, 2013.

[22] S Sicari, A Rizzardi, LA Grieco, and A Coen-Porisini. Security, privacy and trust in internet of things: The road ahead. *Computer Networks*, 76:146–164, 2015.

[23] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.

[24] Xuanxia Yao, Zhi Chen, and Ye Tian. A lightweight attribute-based encryption scheme for the internet of things. *Future Generation Computer Systems*, 49:104–112, 2015.