2017 IEEE 18th International Conference on Mobile Data Management

TrustCEP: Adopting a Trust-Based Approach for Distributed Complex Event Processing

Rahul Dwarakanath[1], Boris Koldehofe[1], Yashas Bharadwaj[2], The An Binh Nguyen[1], David Eyers[3], and Ralf Steinmetz[1]
[1,2]Multimedia Communications Lab (KOM), TU Darmstadt, Germany
firstname.lastname@{[1]kom, [2]stud}.tu-darmstadt.de
[3]Department of Computer Science, University of Otago, New Zealand
dme@cs.otago.ac.nz

***Abstract*—The advent of the Internet of Things (IoT), with modern sensors and sensor-based devices, will significantly stimulate the development of context-aware applications. An effective means to extract higher-level contextual information from sensor data is distributed complex event processing (CEP), which facilitates the analysis of real-time data streams coming from heterogeneous and distributed sources. Considering that user context is inherently sensitive information, the preservation of privacy is critical once the processing of user context takes place over several (possibly malicious) devices, especially in collaborative scenarios.**

In this paper, we tackle this issue by introducing a trust-based approach for the placement and execution of CEP operators in a distributed environment. We propose a trust management model based on communication interactions among the users. Furthermore, we incorporate trust recommendations using a cosine-based similarity check in order to overcome collusion and on-off attacks. We developed a smartphone-based distributed CEP system called *TrustCEP* to evaluate our approach for trust management. Based on the evaluation of TrustCEP, we observe that our approach induces a minimal increase in average battery consumption compared to privacy-negligent approaches.

***Index Terms*—Complex Event Processing, Privacy-aware Operator Placement, Trust Management, D2D Communication**

I. Introduction

With the advent of the Internet of Things (IoT), the paradigm shift towards interconnected devices allows for an improved platform to gather more information about the environment from heterogeneous sources and to exchange information with the real world [1]. This can be especially attributed to the proliferation of sensor-fitted devices (e.g., smartphones) and other sensors in the environment. Consequently, the IoT sets up the pedestal for a renewed form of context-aware computing, where applications interact with the user, and adapt their services based on the prevailing user context.

Complex event processing (CEP) provides a cogent means to discover patterns in event streams (e.g., sensor data), and hence, enables the extraction of higher-level contextual information with high performance and accuracy. Computing modules, often called *operators*, analyze input event streams using functions, such as filtering, and aggregation [2]. Directed, acyclic graphs called *operator graphs* connect the producers (i.e., sources of data) and consumers (i.e., those interested in higher-level context) through event streams, such that they dictate the order in which the operators are to be executed [3].

Recent research has seen a growing interest in the distributed nature of data sources as well as consumers of the information. In one such distributed system, each CEP operator can be placed on one of many potentially capable nodes (e.g., user smartphones), such that different devices take over—through device-to-device (D2D) communication [4], [5]—different parts of the operator graph to collaboratively process the required higher-level contextual information. Generally, the processing of user context entails the analysis of large data volumes coming from heterogeneous sources, necessitating a high amount of resource usage (e.g., battery power, memory space, etc.). Depending on the application at hand, the placement algorithm may need to uphold stringent requirements, such as performance optimization in terms of latency, bandwidth, and energy consumption [6], [7], or high availability and reliability [8].

However, one of the primary aspects that is often neglected during operator placement and the dissemination of events is data privacy. User context is inherently sensitive information—e.g., location, co-location, activity, mood, etc. Thus, data privacy becomes a critical issue once user context needs to be processed in a distributed manner over several (possibly malicious) devices. Having said that, the need for collaboration among the participating devices entails the disclosure of certain possibly sensitive data, to receive the services offered by the supporting application. Therefore, there is a need for a privacy-aware placement mechanism, which adapts event dissemination to the prevailing privacy constraints of the users involved, while still allowing for collaboration.

Let us consider an example scenario to better understand the main problem. Mark and Carl are attending an important release meeting at a business firm. They use a smartphone application, *ContextApp*, to adapt their smartphone experience to their availability based on their context. To recognize the type of meeting, *ContextApp* incorporates the different sound levels—obtained from smartphone microphone sensors—in the meeting room, as well as the location and accelerometer readings of the participants [9]. These raw data are processed on the available user smartphones to obtain the higher-level information (e.g., a calm meeting) using appropriate filtering and aggregation operators, as dictated by rules based on the principles of CEP. Mark is very careful about revealing his sensitive information—microphone data or his mental state—to Carl, fearing that the information may land in unauthorized hands. He fears that his microphone readings in combination with (fine-grained) location or accelerometer information, seen over time, can reveal sensitive information such as his stress level [10], [11]. How can Mark (and the others) process their context in a distributed manner, without revealing their private information to unauthorized users?

To answer this, we propose the incorporation of user trust levels during the placement of CEP operators. Our approach leverages the concept of trust in supply chain management, where the information shared between two users depends on their trust level [12]. By allowing the users to define the sensitivity levels for their generated events and by comparing the trust levels towards other users against the event sensitivity levels, it is possible to control operator placement and thus, event dissemination.

Our main contribution is a (decentralized) trust management model to adapt event dissemination and the placement of operators for distributed CEP. We focus on D2D-based networks, where dynamic changes and resource constraints play a key role. The prime challenges for any trust-based approach is the evaluation and quantification of user trust, and also the robustness of such an approach against attacks. To this end:

- We introduce a trust management model based on user relationships and their communication interaction history;
- We present a robust trust recommendation scheme using the cosine similarity measure that is shown to overcome various attacks on privacy; and
- We implement our trust-based approach within *TrustCEP*, which sets up a D2D-based distributed CEP system, and analyze battery consumption and network data exchange.

Next, we present the exact problem statement in Section II. Section III describes the system fundamentals, focusing on the models and assumptions used in the system. This lays the foundation for our approach for trust-based distributed CEP, described in detail in Section IV. The results for the evaluation of the trust management model as well as TrustCEP are presented and discussed in Section V. Section VI presents some of the existing work related to our research and finally, Section VII concludes the paper and presents future work.

II. Problem Statement

The placement of CEP operators on the appropriate devices is the prerequisite to the execution of an operator graph in a distributed manner. This entails determining the right candidate devices for the sensing and processing tasks. Given that the collaboration of data and processing power among the available devices is necessary for the proper functionality of any context-aware application, it is paramount that the privacy constraints of the users be considered during the (distributed) processing of their (sensitive) sensor data. More specifically, the placement of the CEP operators and the dissemination of the individual events have to be adapted to the privacy constraints posed by the individual sensor data sources (here, primarily, user smartphones). Addressing privacy in distributed CEP systems in D2D-based user environments requires considering the following key aspects.

Event Sensitivity. CEP is largely based on the observation and interpretation of patterns in historical situational information of atomic (e.g., sensor data) and intermediate events. This involves the storage on, and streaming of the events to the respective processing devices. CEP operator graphs may need to access sensitive user information in order to obtain higher-level situational information. Each event can have a different sensitivity level, depending on user privacy constraints. While the final output events (e.g., meeting or not) may not enable an adversary to draw any inference on influential events, the intermediate events (e.g., filtered microphone readings) can allow inference of sensitive user information, e.g., stress level [10], [11]. Therefore, operator placement has to be done in accordance to the user privacy constraints and the corresponding event sensitivity levels.

Presence of Adversaries. Given the inadvertent presence of adversaries, one of the main problems in distributed processing is the difficulty to track the path of the events after their dissemination and monitor the actions of other users [13]. For example, in the motivating scenario, Carl can share Mark's data with any of his friends without Mark's knowledge. Adversaries may share information amongst themselves to infer more sensitive information out of the combination of event streams. Given the lack of system overview in a D2D based network, relying on knowledge gathered from the network does not entirely mitigate these security threats.

Dynamic Environments. The execution of the operator graphs varies with changes in the available sensor data as well as the available processing devices. Each user may have different privacy constraints for operator placement. It is necessary to account for the different user privacy constraints to prevent the unwarranted exposure of sensitive user information to adversaries. Furthermore, the number of adversaries in the network can vary with time. While obfuscation measures (c.f. Schilling et al. [14]) in fixed CEP systems allow for preventing unlawful access to sensitive data, it is a significant challenge to carry out these measures in a scalable manner over multiple devices in a dynamic user environment.

Consequently, the envisaged distributed CEP system must deal with varying user privacy constraints in a possibly hostile environment but still facilitate flexible collaboration amongst the available devices. In our approach, we employ the concept of user trust to facilitate the implementation of one such system. Trust describes the level of confidence a person places in another person for a particular action, without them having the ability to directly control its execution [15], [16]. The management of trust in a distributed environment entails accounting for trust evaluation as well as its evolution with time. In the following sections, we show how we incorporate trust in privacy-aware operator placement for distributed CEP.

III. Preliminaries

Before describing our approach, we provide an overview of the fundamentals behind it, including the main assumptions

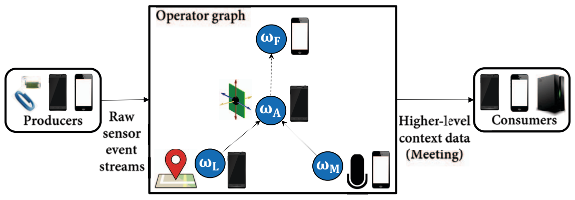


Fig. 1: Illustration of a CEP Operator Graph and System Model

and concepts that play a key role. We shall delve briefly into the CEP model and the proposed system model, including the adversary models considered in our approach.

A. CEP Model

In general, a CEP system constitutes of a set of producers P and consumers C that are connected to each other through a directed, acyclic *operator graph*. Each operator graph comprises a set of correlation operators Ω. The operators are the fundamental computing modules of a CEP system, performing filtering, aggregation, logical, and other rule-based functions. These dictate the path of the events E in the system, such that $E \subseteq (P \cup \Omega) \times (\Omega \cup C)$.

Each operator $\omega \in \Omega$ accepts incoming events I_ω, processes them based on the operator logic and sends the resulting complex events O_ω to the next operator in the operator graph. For example, as seen in Figure 1, the main operators used to satisfy the meeting query are ω_L (to determine the logical location from GPS or Wi-Fi fingerprinting), ω_M (to determine the decibel level in the room based on microphone readings), ω_C (to combine the output events of ω_L and ω_M, and analyze the user movements based on the accelerometer readings), and ω_F (to obtain the final higher-level context based on the output events of ω_C).

For a given operator ω, each (outgoing) event $e \in E$ has a certain *sensitivity* level S_e, indicating how private it is to the corresponding user. We assume that each user sets their event sensitivity levels by themselves based on their privacy constraints and the query at hand. If the trust towards a user A is τ, then event e is disseminated to A iff $\tau \geq S_e$. Accordingly, the corresponding operator ω can be placed on user A's device.

B. System Model

Figure 1 shows an overview of the system model considered in our work. We mainly focus on a system comprising user smartphones and external environmental sensors (see Figure 1), which provide for the sensing and/or processing facilities required for context recognition. We focus on application scenarios where it is more practical to process the information locally amongst the devices, instead of using an external cloud-based service. Such scenarios encourage collaboration amongst the devices in the form of exchanging information and processing power, leading to a more efficient and lower-latency system.

Given the decentralized nature of the information sources, the system model is built around a D2D network that spans a set of users' smartphones. Thus, the CEP operator graphs can be executed in a distributed manner on the available devices. We assume that the devices keep track of other sensing and processing devices in their vicinity, as well as the context queries that the neighboring users are interested in. This is based on the assumption that each user is interested in estimating their own context, so as to optimize the obtained services, accordingly. We assume that there is a consensus beforehand on the user—called the *initiator* (based on, e.g., available battery level or network connectivity)—who initiates the deployment of the operator graph, depending on the context query at hand. We account for the privacy constraints of the initiator as well as the neighboring collaborators to establish a privacy-aware placement of the operators and event dissemination. In turn, we assume that the underlying network is governed by reliable communication primitives, and that all the participating devices are visible to each other at all times (i.e., no hidden node problem).

Furthermore, we assume that the movement of the user devices is dynamic but not highly mobile (i.e., quasi-stationary). For the purpose of our work, we assume that the devices remain available for the entire course of execution of (at least) one operator graph. Additional mechanisms are necessary to deal with dynamic changes by incorporating the migration of CEP operators to appropriate devices during runtime [3], [17].

C. Adversary Model

Distributed systems can face a variety of security/privacy threats. In our work, we focus on *honest-but-curious* adversaries that perform their delegated tasks—the execution of the CEP operators—compliantly, but may try influencing the system to obtain as much sensitive information as possible from the other *benign* users in the environment. Consequently, it is necessary to place the operators only on the authorized devices, such that they may not derive any sensitive information from the transmitted events. Other adversary models can be combated through appropriate security measures, as established in related literature [18].

We consider two types of honest-but-curious attacks: *collusion* attacks where a set of adversaries conspire together to try and manipulate the system functionality to suit their needs; and *on-off* attacks where adversaries behave benignly for a while, gathering trust as they do so, and then turn malicious. Our system is based on the concept of trust levels and recommendations between users (see Section IV). In the context of our work, colluding adversaries tend to manipulate their trust recommendations to improve their reputation in the system. Such adversaries practice collusion by providing either high trust levels (i.e., ballot-stuffing) or low trust levels (i.e., bad-mouthing) to other users.

IV. Trust-Based Distributed CEP

We now present our approach for privacy-aware placement of the CEP operators in a distributed user environment by accounting for the trust levels between the participating users.

We also detail our approach for trust establishment and operator placement in distributed CEP, showing its robustness towards privacy attacks.

A. Approach Overview

Our proposed approach for a privacy-aware adaptation of distributed CEP comprises a trust management model—the evaluation of direct trust between users as well as its evolution—and its incorporation in a privacy-aware operator placement algorithm. The key aspects are:

1) To measure direct trust relations, we build upon the interaction history between the users on synchronous and asynchronous channels, accounting for the behavioral aspects of user trust (see Section IV-B);
2) To enable trust evolution and to reduce the impact of adversaries, we develop an approach that accounts for trust recommendations from other users in the environment (see Section IV-C). This comprises, (i) the exchange of recommendation messages between users, (ii) a similarity-based approach to evaluate the credibility of the received recommendations, and (iii) countermeasures to combat privacy attacks mentioned earlier;
3) In Section IV-D, we describe the algorithm to apply local trust for privacy-aware operator placement.

B. Estimating Direct Trust

It is long established in the field of sociology that the type and strength of a user relationship have a strong correlation with the trust level between the users, and thus with the amount of shared information. Granovetter [19] established the concept of *tie strength*, or the closeness within a user relationship, citing interaction intensity and reciprocity as two main factors. Related research work has since proved that communication characteristics—basically, the interaction history on the synchronous (e.g., calls) and asynchronous (e.g., instant messaging) channels—between two users can be used to estimate their tie strength (and type) [20]–[22], and in turn, their trust level (also termed *behavioral trust* [16], [23]).

We consider the number of synchronous and asynchronous communication events, denoted s and a respectively, as a measure of intensity. For example, the higher the number of calls and/or messages between two users, the higher their trust level. We also consider features which describe distinct usage patterns on the channels. For example, the duration of calls made d and the number of emoticons e in messages are shown to be influential features [21], [22]. Further, we incorporate the degree of reciprocity $\rho \in [0, 1]$ between the users to account for different levels of communication involvement by the users on each channel, denoting the outgoing and incoming events by o and η, respectively (e.g., calls initiated/received). Each feature can have different effects on user trust, and should thus be combined, appropriately. We consider a weight factor $w \in [0, 1]$ for each feature, e.g., $w_{s,o}$ is the weight factor for the number of outgoing synchronous communication events.

For user υ_i, the synchronous and asynchronous components of behavioral trust towards user υ_j are therefore given by the following equations (for sake of clarity, user indices are left out of the equations below).

$$\tau_s = \frac{\rho_s \cdot (w_{s,o}s_o + w_{d,o}d_o) + (1-\rho_s) \cdot (w_{s,\eta}s_\eta + w_{d,\eta}d_\eta)}{s_o + s_\eta + d_o + d_\eta}$$
$$\tau_a = \frac{\rho_a \cdot (w_{a,o}a_o + w_{e,o}e_o) + (1-\rho_a) \cdot (w_{a,\eta}a_\eta + w_{e,\eta}e_\eta)}{a_o + a_\eta + e_o + e_\eta} \tag{1}$$

Therefore, the direct trust value of υ_i towards υ_j is given by,

$$\tau^{\upsilon_i}_{\upsilon_j} = \alpha\ \tau_s + (1-\alpha)\ \tau_a \tag{2}$$

where α balances between the synchronous and asynchronous trust values. If $\Upsilon = (\upsilon_1, \upsilon_2, \ldots, \upsilon_n)$ are the set of n users, we define the trust vector of user υ_i as $T^{\upsilon_i}_{\Upsilon}$, a set of trust levels towards other users $\tau_{\upsilon_j} \forall \upsilon_{j;j\neq i} \in \Upsilon$. The weight parameters—$w$, ρ, and α—may be correlated to each other; the users have the convenience of manipulating the variables by themselves to set up the right trust values. This method facilitates the estimation of the initial trust values.

The obtained normalized trust values range from 0 to 1, with 0 representing complete distrust, 1 for complete trust, and 0.5 for trust neutrality. The interpretation of the trust values is dependent on the application at hand. In general, the farther from 0.5 the trust value lies, the more (un-)trustworthy the corresponding user is.

C. Recommending Trust

The (initial) direct trust $\tau^{\upsilon_i}_{\upsilon_j}$ of user υ_i towards user υ_j is solely based on their previous interaction patterns. However, $\tau^{\upsilon_i}_{\upsilon_j}$ is also dependent on trust recommendations from other users, $\tau^{\upsilon_t}_{\upsilon_j}$ $\forall \upsilon_{t;t\neq i,j}$, especially when υ_i and υ_j have hardly interacted. A lack of trust recommendations minimizes system scope to a limited set of devices and increases the probability of attacks, given the minimal system overview of each user.

In our work, we primarily focus on how a benign user should perceive the received trust recommendations to facilitate privacy-aware distributed CEP among the available devices. In this context, we propose an approach to allow a user to, (i) determine the credibility of the received recommendations, (ii) detect any changes in the behavior of the other users, e.g., from benign to malicious, and (iii) update existing trust values based on the received recommendations.

We address the credibility of received recommendations in the form of a similarity measure, *sim*, based on the cosine distance between the received trust vectors. Furthermore, we propose a conservative approach for modifying the trust values towards other users, where (suspected) adversaries are penalized strongly. We detail each of these aspects in the following.

Incorporating Trust Recommendations. We define trust recommendations as the trust vector $R^{\upsilon_i}_{\Upsilon}$ (not necessarily equal to $T^{\upsilon_i}_{\Upsilon}$) provided by user υ_i to the other users, so that the other users may adapt their trust vectors, accordingly. Each user provides their trust recommendations to other users during each *round*. We define a 'round' as a specific interval in time (e.g., 1 day), which is again application-dependent. The trust recommendation vector of user υ_i, $R^{\upsilon_i}_{\Upsilon}$, sent to user υ_j is modified such that $r^{\upsilon_i}_{\upsilon_{i,j}} = \varnothing$, considering that mutual trust values are private information.

Given that the trust recommendations assist the users in updating trust towards (relatively) unknown users, it is essential to incorporate them appropriately. First and foremost, each user checks the similarity measure, *sim*, of the trust recommendations amongst themselves (to detect colluding users; discussed later), as well as with their own trust values. If similar, the divergence, δ, of the recommended trust values from the current trust values is measured, to indicate if current trust values need to be adjusted. To this end, we employ a conservative additive increase, multiplicative decrease principle (similar to that used for TCP congestion control) for trust updates, depending on the valency of the divergence. This allows the benign users to improve the trust evaluation of the other users, and also mitigate the effects of adversaries.

Trust Similarity Check. The measure of similarity allows a user to determine two main aspects about trust: (1) The credibility of the recommendations of other users, and (2) the level of trust they can assign to (relatively) unknown users. In our approach, each user first considers the trust recommendation vectors sent by the highly trusted users. Further, each user uses the cosine similarity measure between the remaining trust recommendation vectors and their own trust vector to find out the *semantically* similar recommendations [24]. The similarity measure, $sim^{\upsilon_i}_{\upsilon_j} \in [0, 1]$ between the trust vector of υ_i, $T^{\upsilon_i}_{\Upsilon}$, and the recommended trust vector of υ_j, $R^{\upsilon_j}_{\Upsilon}$, is given by the cosine of the angle between the two vectors, as in (3). This indicates the extent of divergence between trust vectors.

$$sim^{\upsilon_i}_{\upsilon_j} = \frac{T^{\upsilon_i}_{\Upsilon} \cdot R^{\upsilon_j}_{\Upsilon}}{|T^{\upsilon_i}_{\Upsilon}| \cdot |R^{\upsilon_j}_{\Upsilon}|} = \frac{\sum_{t=1;t\neq i,j}^{n} \tau^{\upsilon_i}_{\upsilon_t} \cdot r^{\upsilon_j}_{\upsilon_t}}{\sqrt{\sum_{t=1;t\neq i}^{n} (\tau^{\upsilon_i}_{\upsilon_t})^2} \cdot \sqrt{\sum_{t=1;t\neq j}^{n} (r^{\upsilon_j}_{\upsilon_t})^2}} \tag{3}$$

Although mathematically inconsequential, the dot product is not applied for $t = i, j$ due to the semantics of trust recommendation mentioned above. The premise behind the cosine-based approach is the relative proclivity of a user towards the other users. Even if a user has different ways to estimate their direct trust towards the other users, leading to different trust values, the relative proclivity of the user determines how comparable these values are. For example, if user υ_i's trust vector contains $T^{\upsilon_i}_{\Upsilon} = [0.4; 0.8]$ and user υ_j recommends the trust values $R^{\upsilon_j}_{\Upsilon} = [0.3; 0.7]$, then both the users have a similar proclivity towards the others but different trust estimation weights, as seen in the small angle between these two vectors, with $\cos^{-1}(sim^{\upsilon_i}_{\upsilon_j}) = 3.82°$.

In turn, each user measures the mean divergence (displacement) of the recommended trust values from their own. The difference of the recommended trust values from the user trust values is passed through a weighting function, dependent on the similarity measure with each recommender, to obtain the trust divergence (4). If $|\delta^{\upsilon_i}_{\upsilon_j}| \geq \delta_{threshold}$, then the corresponding trust value $\tau^{\upsilon_i}_{\upsilon_j}$ is modified based on the trust increase/decrease coefficients τ_{inc} and τ_{dec}, as in (5).

Algorithm 1 Combating Privacy Attacks (w.r.t. User υ_i)

```
1:  $T^{\upsilon_i}_{\Upsilon} \leftarrow$ current trust vector of $\upsilon_i$
2:  $markedUsers, sim_R, recHist \leftarrow \varnothing$
3:  for $roundNumber = 1$ to $N$ do
4:      for $j = 1$ to $n$ and $j \neq i$ do
5:          function $rcvRecommendations(R^{\upsilon_j}_{\Upsilon}) \ni r^{\upsilon_j}_{\upsilon_{i,j}} = \varnothing$
6:              if $calcSim(R^{\upsilon_j}_{\Upsilon}, recHist[j]) < sim_{threshold}$ then
7:                  if $roundNumber > 1$ then
8:                      $markedUsers[j] \mathrel{+}= 1$
9:                  end if
10:             else if $markedUsers[j] < mal_{threshold}$ then
11:                 $sim^{\upsilon_i}_{\upsilon_j} \leftarrow calcSim(T^{\upsilon_i}_{\Upsilon}, R^{\upsilon_j}_{\Upsilon})$
12:             end if
13:             $recHist[j] \leftarrow R^{\upsilon_j}_{\Upsilon}$
14:         end function
15:     end for
16:     for $j = 1$ to $n$ and $j \neq i$ do
17:         for $k = 1$ to $n$ and $k \neq i$ do
18:             $sim_R[j] \mathrel{+}= calcSim(R^{\upsilon_j}_{\Upsilon}, R^{\upsilon_k}_{\Upsilon})$
19:         end for
20:     end for
21:     for $j = 1$ to $n$ and $j \neq i$ do
22:         if $|(\mu_{sim_R} - sim_R[j])| > \sigma_{sim_R}$ then
23:             $markedUsers[j] \mathrel{+}= 1$
24:         end if
25:     end for
26:     if $roundNumber > N_{wait}$ then
27:         for all $j \in markedUsers < mal_{threshold}$ do
28:             $markedUsers[j] \mathrel{-}= 1$    // loyalty-based trust
29:             $\tau^{\upsilon_i}_{\upsilon_j} \mathrel{+}= \tau_{inc} \ni \max(\tau) = 1$
30:             $modifyTrustBasedOnDivergence(T^{\upsilon_i}_{\Upsilon}, sim)$
31:         end for
32:         for all $j \in markedUsers > mal_{threshold}$ do
33:             $\tau^{\upsilon_i}_{\upsilon_j} \mathrel{/}= \tau_{dec}$
34:         end for
35:     end if
36: end for
```

$$\delta^{\upsilon_i}_{\upsilon_j} = \frac{\sum_{t=1;t\neq i,j}^{n} \sigma^{\upsilon_i}_{\upsilon_t} \cdot (\tau^{\upsilon_i}_{\upsilon_j} - r^{\upsilon_t}_{\upsilon_j})}{\sum_{t=1;t\neq i,j}^{n} sim^{\upsilon_i}_{\upsilon_t}} \tag{4}$$

$$\tau^{\upsilon_i}_{\upsilon_j} = \begin{cases} \tau^{\upsilon_i}_{\upsilon_j} + \tau_{inc}, & \text{if } \delta^{\upsilon_i}_{\upsilon_j} < 0 \\ \tau^{\upsilon_i}_{\upsilon_j} / \tau_{dec}, & \text{otherwise} \end{cases} \ni |\delta^{\upsilon_i}_{\upsilon_j}| \geq \delta_{threshold} \tag{5}$$

Combating Privacy Attacks. Apart from the above approach to modify the trust vectors, each user should also penalize the users that provide falsified trust values. Given the adversary models in Section III, we now describe how our approach deals with such privacy attacks (see Algorithm 1).

Collusion Attacks: In our work, adversaries try to manipulate the system by colluding with other adversaries and sending falsified trust recommendations to other users. To combat such attacks, each user analyzes the similarity among the incoming trust recommendations and detects the users that provide uncorrelated values (Line 23). Such recommendations are discarded and, to thwart such future attempts, in each round, these users are marked as adversaries. Rather than risk marking possibly benign users, each user waits a few rounds, N_{wait}, before analyzing the behavior of the other users. If the number of marks exceeds the *malicious threshold*, $mal_{threshold}$, the corresponding users are considered malicious and their trust levels are decreased (Line 34). A large value for $mal_{threshold}$ will leave adversaries undetected for longer periods of time; a small value may classify benign users as malicious. Conversely, each user also employs a loyalty-based trust model, where the trust value of the users who remain non-malicious (below $mal_{threshold}$) for the waiting period, N_{wait}, is increased (Line 30). This allows for the increase of the trust value towards falsely-labelled benign users.

On-off Attacks: In our adversary model, adversaries can behave benignly for a number of rounds, N_{benign}, before turning malicious. This intermittent malicious behavior can be very hard to handle, especially when the occurrence of these aberrations is frequent. While there have been dedicated ways to prevent such attacks proposed in related literature [25], we mainly deal with such attacks by enforcing a high penalty on adversaries (τ_{dec} in (5)). Aberrations in trust recommendations are handled by checking the similarity between past recommendations (Lines 7–9). If $sim < sim_{threshold}$, these users are marked and handled just as described above.

D. Trust-based Operator Placement

Algorithm 2 describes our approach for trust-based placement of CEP operator graphs. Initially, user v_i searches for neighboring users and sets up the trust vector $T^{v_i}_{\Upsilon}$ based on the interaction history as well as current trust recommendations R_{Υ} from other users (see Algorithm 1). Based on the context query at hand, the *initiator* (see Section III) sets up the event sensitivities with respect to their privacy constraints and creates the required operator graph (Lines 7–9; c.f. [2], [3]).

If no neighboring users are detected, the operator graph is executed on the initiator's own device. Otherwise, the initiator sets up the placement graph (Line 13) based on the trust vector and the received recommendations, and delegates the operators to the collaborating users by sending placement requests. Upon reception of a request message, the users check if the request violates their privacy constraints based on their trust vector as well as their event sensitivity values. If so, they send a conflict message C_{Υ} back to the initiator and notify, if available, any substitute user who can take over the conflicted path (Line 18). The initiator then modifies the placement graph and resends the new requests. Once the trust negotiations are complete, the initiator starts the operator graph execution.

Algorithm 2 Trust-Based Operator Placement (v_i as Initiator)

```
1: $\Upsilon \leftarrow findNeighboringUsers()$
2: if $\Upsilon \neq \varnothing$ then
3:     $T^{v_i}_{\Upsilon} \leftarrow evaluateDirectTrust(\Upsilon)$
4:     $R_{\Upsilon} \leftarrow obtainTrustRecommendations(\Upsilon)$
5:     $T^{v_i}_{\Upsilon} \leftarrow modifyTrustValues(T^{v_i}_{\Upsilon}, R_{\Upsilon})$
6: end if
7: function $rcvQuery(Q)$
8:     $S \leftarrow setupEventSensitivity(Q)$
9:     $O \leftarrow establishOperatorGraph(Q, S)$
10:    if $\Upsilon = \varnothing$ then
11:        $executeOperatorGraph(O)$
12:    else
13:        $P^{v_i}_{\Upsilon} \leftarrow establishPlacementGraph(O, T^{v_i}_{\Upsilon}, R_{\Upsilon})$
14:        $sendPlacementRequests(P^{v_i}_{\Upsilon}, \Upsilon)$
15:    end if
16: end function
17: function $rcvResponse(\Upsilon, C_{\Upsilon})$
18:    if $C_{\Upsilon} \neq \varnothing$ {// trust conflict} then
19:        $P^{v_i}_{\Upsilon} \leftarrow modifyPlacementGraph(P^{v_i}_{\Upsilon}, C_{\Upsilon})$
20:        $sendPlacementRequests(P^{v_i}_{\Upsilon}, \Upsilon)$
21:    else
22:        $executeOperatorGraph(P^{v_i}_{\Upsilon})$
23:    end if
24: end function
```

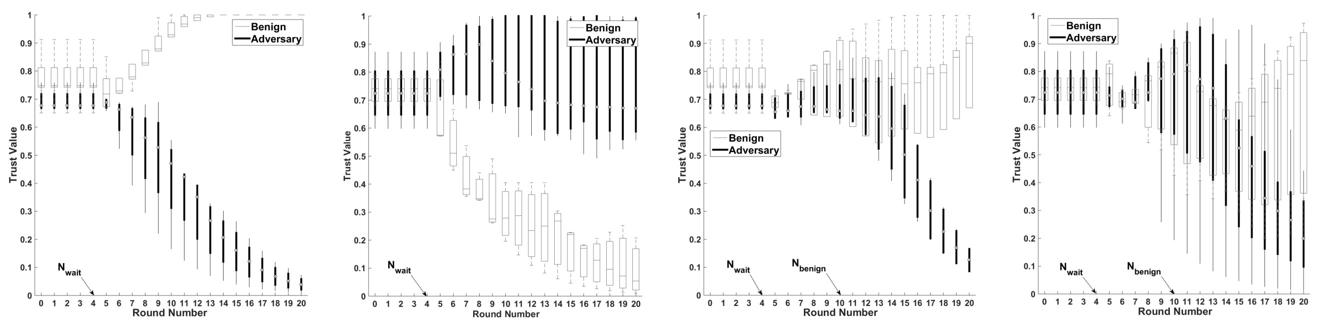
V. Evaluation Setup and Results

The evaluation of our proposed approach comprises two parts—the analysis of the proposed trust management model in terms of its efficacy, and the evaluation of its applicability in a distributed CEP system for privacy-aware operator placement. We analyze the efficacy of the trust management model by simulating the behavior of the adversaries, depending on the type of attacks, and observing the changes to the (average) trust level towards the users in the system. To evaluate the applicability of the proposed trust-based approach, we designed a smartphone-based distributed CEP system called *TrustCEP* that allows the users to communicate with each other using Bluetooth and process the operator graphs in a distributed manner. In turn, we measure the battery consumption incurred upon the smartphones as well as network data exchange, by deploying our trust-based approach for collaborative processing. In the following, we detail the test cases for each evaluation step and discuss the results obtained.

A. Evaluation of Trust Management Model

First, we evaluate our trust management model, focusing on collusion and on-off attacks. For the collusion attacks, the adversaries adjust the trust values by increasing/decreasing them by 0.5 subject to $\tau \in [0, 1]$. For the on-off attacks, the adversaries behave benignly for N_{benign} rounds before starting to collude. The initial trust vectors of each user were generated randomly between 0.5 and 1 using a uniform distribution function. We ran each of the simulations for a fixed number of rounds and observed the trust levels held by benign users towards the other users. Table I shows the default values of the system parameters used in our experiments.

Case 1: Collusion Attacks. Figures 2a and 2b show the results of the collusion attacks analysis. The box plots provide the median as well as the 25th and 75th percentiles of the trust



(a) Collusion Attack Analysis: 10 Users; 4 Adversaries; 6 Benign

(b) Collusion Attack Analysis: 10 Users; 7 Adversaries; 3 Benign

(c) On-off Attack Analysis: 10 Users; 4 Adversaries; 6 Benign

(d) On-off Attack Analysis: 10 Users; 7 Adversaries; 3 Benign

Fig. 2: Trust Evaluation Analysis for Collusion and On-off Attacks

value distribution over the different rounds. Figure 2a shows the distribution of the trust values towards the benign users and the adversaries, with 40% adversaries. We notice a clear decrease in the trust value obtained by adversaries after N_{wait}, with 75% of the adversaries below trust neutrality by round 10. The variance in their trust value distribution also reduces with time, given the multiplicative trust reduction. Further, we observe that the trust value obtained by the benign users increases, primarily due to the loyalty-based trust model.

Contrarily, in Figure 2b, we observe that the high number of adversaries (70%) leads to a converse effect to the trust values. While over 50% of the adversaries obtain a near-average trust value of 0.7 or lower, the trust value obtained by the benign users decreases considerably. This can be attributed to the similarity-based analysis, where the benign users are assumed to be providing falsified recommendations. It should be noted that this phenomenon occurs primarily when the adversaries provide falsified recommendations to *all* benign users.

Case 2: On-off Attacks. For the on-off attack analysis, we used the same initial trust values as in each case for the collusion attack analysis. Here, unlike above, the adversaries start colluding after $N_{benign} = 10$ rounds and provide falsified trust recommendations to only a (randomly) selected set of benign users. We observe in Figure 2c, for the case with 40% adversaries, the trust value obtained by the adversaries decreases rapidly after they switch behavior. The converse is observed in the trust value obtained by benign users, where their trust value improves after a few rounds, despite being negatively affected in the beginning. In Figure 2d, we observe that the trust value obtained by the adversaries decreases below trust neutrality 6 rounds after N_{benign}. This attributes to the above-mentioned fact that the trust recommendations of the adversaries do not reach *all* benign users, facilitating their detection based on the similarity measure, sim. We also observe that the on-off nature of the adversaries affects the trust values obtained by the benign users, leading to a longer delay in obtaining the true trust values.

Case 3: Waiting Period. We also analyzed the effects of the waiting period, N_{wait}, in case of on-off attacks with 40% adversaries. In Figure 3a, with $N_{wait} = 0$, we observe that the adversaries obtain a very high trust value before their true identities are uncovered ($N_{benign} = 10$). Furthermore, some benign users are also marked as malicious, resulting in a lower trust value towards them. In Figure 3b, if $N_{wait} = 10$, we observe that the adversaries obtain a lower trust value immediately after switching behavior. However, we observe that the benign users obtain a relatively mixed trust value, which is attributed to the longer period of non-malicious behavior necessary for loyalty-based trust increase (see Algorithm 1).

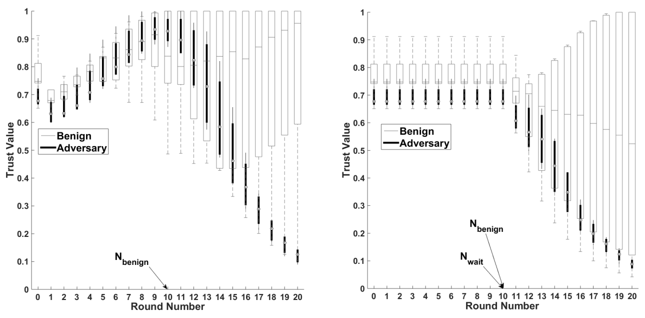
TABLE I: Default Parameters for Trust Evaluation

Parameter	Value
No. of users, n	10
% Adversaries	$\underline{40}$, 70
Similarity threshold, $sim_{threshold}$	0.95
Divergence threshold, $\delta_{threshold}$	0.1
Malicious threshold, $mal_{threshold}$	5
Trust increase coefficient, τ_{inc}	0.5
Trust decrease coefficient, τ_{dec}	0.75

Discussion. We observe that our approach combats collusion and on-off attacks when the adversaries are in the minority. Even in case of a majority, the adversaries can be detected if they do not infiltrate the entire network. Overall, τ_{inc} and τ_{dec} should be chosen appropriately, such that the higher trust levels are achieved only after substantial amount of time. It is necessary to adjust the event sensitivity levels accordingly, so that only the highly trusted receive sensitive data. Since adversaries are only detected after a few rounds, it is advisable to wait a few rounds before employing the trust values for operator placement. This can also be addressed by adapting the application logic, accordingly. For example, in the motivating scenario, location or microphone events primarily become sensitive after certain rounds of operation due to the presence of historic information. By adapting the event sensitivity levels and/or by randomizing the event recipients in each round, one can avoid serious privacy violations.

B. Evaluation of TrustCEP

We designed TrustCEP to understand the applicability and practicability of our trust-based approach in a real-life distributed CEP system. The conceptual design of our prototypical simulation environment concurs with modern research to-



(a) 10 Users; 4 Adversaries; 6 Benign; $N_{wait} = 0$ (b) 10 Users; 4 Adversaries; 6 Benign; $N_{wait} = 10$

Fig. 3: Trust Evaluation for Different Waiting Periods w.r.t. On-off Attacks

wards D2D communication as part of the IoT [4], [5], such that users can collaboratively extract contextual information from the environment based on their needs. We chose Bluetooth as the basis for D2D communication, given its ability to support low-power data transmission (compared to other standards like Wi-Fi Direct) and its ubiquitous availability [26]. For the prototypical evaluation, we used five Google Nexus 5 smartphones (with Bluetooth 4.0), running Android 4.3 or higher. To allow for Bluetooth communication, all the detected smartphone devices were paired in advance.

Initially, TrustCEP allows users to set up the event sensitivity levels and the various weights needed for trust estimation beforehand. For the evaluation, we generated the interaction history between the users randomly, including users with high and low communication intensity, and a few without any (corresponding to trust neutrality). Based on the motivating scenario, we consider low-level (atomic) events coming from microphone, GPS, and accelerometer readings. Considering that these atomic events are noise-ridden, they are each first processed by a filtering operator. The filtered higher-level events are then processed using additional operators (aggregation, composition, or derivation) as part of an operator graph, as described in Section III (c.f. Figure 1). In total, we consider 7 operators in the operator graph. All events are modeled as event objects, and appended with appropriate attributes for further processing (e.g., time stamp, source node ID).

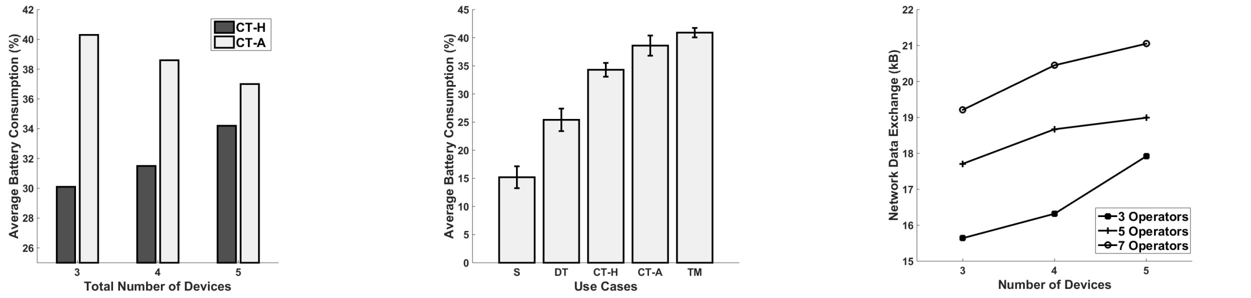
Our main goal is to understand the effects of our trust-based approach on battery consumption and network data exchange. For measuring battery consumption, we charge all devices up to 100% beforehand and note down the end battery level after 100 rounds. All other communication services are blocked during the evaluation period. We focus on the battery consumption at the initiator, given that the operator graph deployment is primarily dependent on the trust management model of the initiator. We compare our approach against collaborative approaches without a trust model as well as non-collaborative approaches, based on the following use cases. Note that we speak of devices as an extension of the respective users and their privacy constraints.

1) **Use Case *S*:** All the operators in the operator graph are executed on a single device.
2) **Use Case *DT*:** The devices try discovering other devices but distrust all of them, and thus, execute the entire operator graph themselves (without the help of the others).
3) **Use Case *CT-A*:** The devices have absolute trust towards each other. Here, the *initiator* primarily analyzes the atomic events, while the higher level events are analyzed by the other collaborating devices. Note that, as described in Section IV-D, the *initiator* deploys the operator graph among the available devices (including itself).
4) **Use Case *CT-H*:** Similar to use case **CT-A**. However, here, the *initiator* primarily analyzes the high-level events, and the other devices analyze the atomic events.
5) **Use Case *TM*:** Here, the devices trust each other to different extents and collaborate to execute operators in the operator graph based on the proposed trust management model. For the sake of simplicity, we do not consider the presence of any adversaries in this analysis.

Varying the Number of Devices. To understand the influence of multiple device collaboration on battery consumption, we executed the complete trust use cases **CT-A** and **CT-H** for 100 rounds on 3 to 5 devices, by adjusting the trust levels on all devices to the maximum. The results in Figure 4a show that the analysis of the atomic events is battery-heavier than the analysis of higher-level events. The average battery consumption at the initiator in use case **CT-H** increases with increase in the number of collaborating devices. This can be attributed to the increase in number of control messages exchanged between the devices for collaboration, depending on the placement of the operators. Also, the average battery consumption at the initiator decreases in use case **CT-A**, primarily due to the availability of more processing devices and the delegation of high-level operators to them.

Analysis of the Use Cases. We analyzed the battery performance at the initiator for all use cases with 5 devices; Figure 4b shows the average battery consumption for all devices acting as an initiator in each use case. We observe that the battery consumption is the least in use case **S**, where there is no communication among the devices. This acts as the baseline for our analysis of the collaborative scenarios. For use case **DT**, we adjusted the trust values on all devices to the minimum. Consequently, comparing to use case **S**, we observe that, on average, 10.2% of the battery is spent on discovering Bluetooth devices in the neighborhood. The battery consumption in use case **TM** is marginally higher than that in use cases **CT-A** and **CT-H**, by around 2% and 6%, respectively. This is mainly attributed to the recommendation and trust negotiation messages exchanged between the devices.

Network Data Analysis. Finally, we analyze the number of bytes exchanged between devices in use case **TM**, for different number of available devices as well as different number of operators in the operator graph (Figure 4c). In case of 5 operators, only microphone and location events were considered; for 3 operators, only microphone events were considered. We observe that there is an increase in the number of bytes exchanged with increase in the number of operators



(a) Average Battery Consumption at Initiator over 100 Rounds with Increase in Number of Devices

(b) Average Battery Consumption at Initiator over 100 Rounds for All Cases

(c) Average Bytes Exchanged per Round under Varying Number of Operators and Devices

Fig. 4: Evaluation Results of TrustCEP in Terms of Battery Consumption and Network Data Exchange

as well as the number of available devices. The additional number of messages can be attributed to the increase in control messages (recommendations, trust negotiations, operator placement messages) as well as the different number and size of data messages for the different atomic events. The use of Bluetooth for transmitting the considerably small-sized event and control messages contributes to the minimal additional battery consumption using the trust management model.

Discussion. From the above analysis, we can ascertain that TrustCEP is not practicable for applications, where user context can be extracted from the sensor data available on a single device, considering that the battery consumption for single device execution is considerably lower (see Figure 4b). However, for context-aware applications in the IoT with decentralized sources of sensor data, TrustCEP allows for a feasible privacy-aware collaboration amongst devices, consuming less than 0.5% of smartphone battery for a complete execution cycle (here, an operator graph). Of course, one such operator placement approach must also consider other factors including the resource availability and mobility patterns of the available devices, apart from the privacy constraints of the users.

VI. Related Work

Most existing approaches for distributed event processing and operator placement neglect privacy altogether. Instead, they concentrate on system optimization with respect to network performance metrics, such as latency, bandwidth, and network load [6]. For example, Ottenwälder et al. [3] consider the case of latency for the placement of CEP operators in the scope of mobile situation awareness (MSA) applications. Towards energy-efficiency, Yang et al. [7] devise the CQP framework for collaborative processing of complex queries among user mobile devices, avoiding processing overhead due to redundant data transmission. Thus, they reduce the average energy consumption in the system. Starks et al. [27] present an energy-efficient mechanism for distributed CEP in MANETs, minimizing the energy consumption in resource-constrained devices by limiting the data transmission costs and achieving near-optimal operator placement.

The existing approaches towards privacy-preserving CEP (PP-CEP) predominantly lay focus on static operator graphs and/or scenarios with a low degree of dynamics. He et al. [28] present a concept for PP-CEP by analyzing the implications of event pattern reporting on user privacy. By grouping the generated event patterns into public and private patterns—those that should and should not be reported, respectively—they propose a mathematical foundation for PP-CEP such that private patterns can be suppressed intelligently without compromising on the utility of the CEP system. Schilling et al. [14] approach this issue by considering user access policies that are used to obfuscate private event patterns over a chain of dependent CEP operators. They mainly focus on developing a scalable method to measure the obfuscation imposed by the access policies. The above approaches, however, are not feasible in dynamic scenarios where the *a priori* knowledge of the producers, consumers, and intermediate processing devices is not available. Furthermore, these approaches can be used in parallel with ours, where we identify the trustworthiness of the collaborating devices to establish the user access policies.

Trust has played a key role in pervasive and peer-to-peer computing, generally based on the exchange of services between devices and the quality experienced thereof [16]. Vidyalakshmi et al. [29] introduce a decentralized trust model for access control in content sharing among mobile devices. They build trust models based on categories and contexts for files along with their perceived sensitivity, thus enabling a more flexible way of access control. Quercia et al. [15] put forth a computational trust framework for pervasive computing based on Bayesian formalization, by incorporating trust dimensions of subjectiveness, time, and context.

Adversaries have been combated in different ways in trust-related literature. Srivatsa et al. [30] propose a reputation management system by incorporating historical reputations and behavioral changes. To deal with colluding adversaries, they propose a personalized similarity measure that accounts for trust variance between two users. However, unlike our approach, they do not account for the cold start problem, where there is a lack of interaction between users. Sun et al. [31] put forth a trust modeling framework for ad-hoc networks, considering trust as a measure of uncertainty. Their main focus lies in facilitating efficient packet routing using the concept of trust propagation. Their approach separates action trust from recommendation trust, and mainly overcomes bad-mouthing and Sybil attacks. Das and Islam [32] deal with oscillating adversary behavior in multiagent systems by introducing a feedback-based dynamic trust computation model. Their approach is similar to ours but suffers from infeasible computational delays, given the large number of trust-related parameters to be evaluated.

VII. Summary and Future Work

The upcoming revolutionary technologies such as the IoT and D2D communication provide for a promising future for context-aware applications. Considering that user context is inherently sensitive information, the preservation of data privacy is a key challenge to be addressed for such applications. Current work on distributed complex event processing (CEP) often neglects the effects of data privacy. To this end, we proposed a trust-based approach for distributed CEP which leverages the trust between users based on their interaction history as well as user recommendations. Through experimental results, we show that our trust management model is robust towards collusion and on-off attacks when the adversaries are in the minority. We implemented our approach within TrustCEP, allowing for collaborative processing of CEP operator graphs using D2D communication between smartphones. Based on its evaluation, we assert its feasibility to enable privacy-aware operator placement, with a negligible increase of 2–6% in battery consumption compared to privacy-negligent approaches.

As part of future work, we plan to apply our approach in more dynamic and mobile environments where the migration of CEP operators may be necessary. Operator migration can be performed by transferring selected parts of an operator and the associated events to a more capable device. Adapting user access policies to such fine-granular cases can have a promising impact on future context-aware applications. Furthermore, we plan to evaluate the usability and practicability of our trust management model as part of a user study.

Acknowledgment

This work has been [co-]funded by the Social Link and NICER projects within the Loewe Program of Excellence in Research, Hesse, Germany, and by the German Research Foundation (DFG) as part of the project C2 within the Collaborative Research Centre (CRC) 1053 – MAKI.

References

[1] A. Zanella *et al.*, “Internet of Things for Smart Cities,” *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.

[2] G. Cugola and A. Margara, “Deployment Strategies for Distributed Complex Event Processing,” *Computing*, vol. 95, no. 2, pp. 129–156, 2013.

[3] B. Ottenwälder *et al.*, “MCEP: A Mobility-Aware Complex Event Processing System,” *ACM Transactions on Internet Technology*, vol. 14, no. 1, pp. 6:1–6:24, 2014.

[4] O. Bello and S. Zeadally, “Intelligent Device-to-Device Communication in the Internet of Things,” *IEEE Systems Journal*, vol. 10, no. 3, pp. 1172–1182, 2014.

[5] L. Militano *et al.*, “Device-to-Device Communications for 5G Internet of Things,” *EAI Endorsed Transactions on Internet of Things*, vol. 15, no. 1, pp. 1–15, 2015.

[6] G. T. Lakshmanan, Y. Li, and R. Strom, “Placement Strategies for Internet-Scale Data Stream Systems,” *IEEE Internet Computing*, vol. 12, no. 6, pp. 50–60, 2008.

[7] J. Yang *et al.*, “Energy-Efficient Collaborative Query Processing Framework for Mobile Sensing Services,” in *IEEE MDM*, 2013, pp. 147–156.

[8] B. Koldehofe *et al.*, “Rollback-Recovery without Checkpoints in Distributed Event Processing Systems,” in *ACM DEBS*, 2013, pp. 27–38.

[9] S. W. Loke, “On Representing Situations for Context-Aware Pervasive Computing: Six Ways to Tell if You Are in a Meeting,” in *IEEE PerCom Workshops*, 2006, pp. 35–39.

[10] H. Lu *et al.*, “Stresssense: Detecting Stress in Unconstrained Acoustic Environments Using Smartphones,” in *ACM Ubicomp*, 2012, pp. 351–360.

[11] A. Sano and R. W. Picard, “Stress Recognition Using Wearable Sensors and Mobile Phones,” in *IEEE ACII*, 2013, pp. 671–676.

[12] N. Ebrahim-Khanjari, W. Hopp, and S. M. Iravani, “Trust and Information Sharing in Supply Chains,” *Production and Operations Management*, vol. 21, no. 3, pp. 444–464, 2012.

[13] J. Singh *et al.*, “Big Ideas Paper: Policy-Driven Middleware for a Legally-Compliant Internet of Things,” in *ACM/IFIP/USENIX Middleware*, 2016, pp. 13:1–13:15.

[14] B. Schilling, B. Koldehofe, K. Rothermel, and U. Ramachandran, “Access Policy Consolidation for Event Processing Systems,” in *IEEE NetSys*, 2013, pp. 92–101.

[15] D. Quercia, S. Hailes, and L. Capra, “B-trust: Bayesian Trust Framework for Pervasive Computing,” in *IFIP TM*, 2006, pp. 298–312.

[16] W. Sherchan, S. Nepal, and C. Paris, “A Survey of Trust in Social Networks,” *ACM Computing Surveys*, vol. 45, no. 4, pp. 47:1–47:33, 2013.

[17] R. Dwarakanath, B. Koldehofe, and R. Steinmetz, “Operator Migration for Distributed Complex Event Processing in Device-to-Device Based Networks,” in *ACM M4IoT*, 2016, pp. 13–18.

[18] M. La Polla, F. Martinelli, and D. Sgandurra, “A Survey on Security for Mobile Devices,” *IEEE Communications Surveys & Tutorials*, vol. 15, no. 1, pp. 446–471, 2013.

[19] M. Granovetter, “The Strength of Weak Ties: A Network Theory Revisited,” *Sociological theory*, pp. 201–233, 1983.

[20] E. Gilbert and K. Karahalios, “Predicting Tie Strength with Social Media,” in *ACM CHI*, 2009, pp. 211–220.

[21] J. Wiese, J.-K. Min, J. I. Hong, and J. Zimmerman, “You Never Call, You Never Write: Call and SMS Logs Do Not Always Indicate Tie Strength,” in *ACM CSCW*, 2015, pp. 765–774.

[22] R. Dwarakanath *et al.*, “Analyzing the Influence of Instant Messaging on User Relationship Estimation,” in *IEEE MS*, 2016, pp. 49–56.

[23] S. Adali *et al.*, “Measuring Behavioral Trust in Social Networks,” in *IEEE ISI*, 2010, pp. 150–152.

[24] S. Zhao *et al.*, “Improved Recommendation Based on Collaborative Tagging Behaviors,” in *ACM IUI*, 2008, pp. 413–416.

[25] Y. Chae, L. C. DiPippo, and Y. L. Sun, “Trust Management for Defending On-off Attacks,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 4, pp. 1178–1191, 2015.

[26] R. Friedman, A. Kogan, and Y. Krivolapov, “On Power and Throughput Tradeoffs of WiFi and Bluetooth in Smartphones,” *IEEE Transactions on Mobile Computing*, vol. 12, no. 7, pp. 1363–1376, 2013.

[27] F. Starks and T. P. Plagemann, “Operator Placement for Efficient Distributed Complex Event Processing in MANETs,” in *IEEE WiMob Workshop*, 2015, pp. 83–90.

[28] Y. He, S. Barman, D. Wang, and J. F. Naughton, “On the Complexity of Privacy-Preserving Complex Event Processing,” in *ACM SIGMOD/PODS*, 2011, pp. 165–174.

[29] B. Vidyalakshmi, R. K. Wong, and C.-H. Chi, “Decentralized Trust Driven Access Control for Mobile Content Sharing,” in *IEEE Big Data*, 2013, pp. 239–246.

[30] M. Srivatsa, L. Xiong, and L. Liu, “TrustGuard: Countering Vulnerabilities in Reputation Management for Decentralized Overlay Networks,” in *WWW*, 2005, pp. 422–431.

[31] Y. L. Sun, W. Yu, Z. Han, and K. R. Liu, “Information Theoretic Framework of Trust Modeling and Evaluation for Ad Hoc Networks,” *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 2, pp. 305–317, 2006.

[32] A. Das and M. M. Islam, “SecuredTrust: A Dynamic Trust Computation Model for Secured Communication in Multiagent Systems,” *IEEE Transactions on Dependable and Secure Computing*, vol. 9, no. 2, pp. 261–274, 2012.