IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 21, 2019, accepted February 12, 2019, date of publication February 22, 2019, date of current version March 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2901023

Trust Management of Smart Service Communities

HAMID AL-HAMADI[1], **ING-RAY CHEN**[2], **AND JIN-HEE CHO**[2], **(Senior Member, IEEE)**
[1]Department of Computer Science, Kuwait University, Safat 13060, Kuwait
[2]Department of Computer Science, Virginia Tech, Blacksburg, VA 24061, USA

Corresponding author: Hamid Al-Hamadi (hamid@cs.ku.edu.kw)

This work was supported by the Kuwait University Research Grant QS01/18.

ABSTRACT In this paper, the notion of a smart service community is proposed to address the grand challenge of a huge number of the Internet-of-Things (IoT) devices providing similar services in a smart city environment (e.g., parking, food, healthcare, transportation, and entertainment). We propose that a smart service community be built as a cloud utility accessible via a mobile application installed in user-owned IoT devices, such as smartphones. The cloud utility provides cloud-based interfaces, including registration, service satisfaction reporting, recommender credibility reporting, and service recommendation, with the goal of recommending the best service providers based on a user's specified service performance criteria. The trust-based service management techniques, utilizing the IoT-assisted technology, are developed to automatically measure service ratings and recommender credibility ratings, and compute one-to-one subjective trust scores to allow a user to select the best service providers among all. The feasibility of the proposed approach is demonstrated over contemporary service ranking systems using a smart food service community for which the major performance metric is the service wait time.

INDEX TERMS Smart service community, trust management, collusion attack, smart city, Internet-of-Things (IoT), service management.

I. INTRODUCTION

A grand challenge in a future smart city or smart world environment is that there will be a huge number of IoT devices providing similar services (e.g., parking, food, healthcare, transportation, and entertainment). Numerous examples can be found in our day-to-day life that we are constantly trying to find the best service provider (SP) for a specific service [1]–[3]. When there are many similar SPs available, a service requester (SR) will naturally try to determine the best one available. A common method of determining the best SP is to compare them based on service ratings posted in a service community application, such as *Yelp* for food service, which links to SP advertisements about the service provided. This process is not only time consuming but also vulnerable to ballot-stuffing attacks (i.e., saying a bad SP is a good SP) or bad-mouthing attacks (i.e., saying a good SP is a bad SP) by which the rating is boosted or defamed by malicious raters.

Very frequently every user has its own idea of the most critical metrics. For example, for food service, a user may think the most critical metric is whether an SP can provide the service earliest to the customer. In other words, if multiple SPs are available with a similar quality of a service, then the customer will select the one with the least time to wait. *Google Maps*[4] is an example of an application enabling a user to view both the average service time and the average wait time at a restaurant, where the average service time is estimated by automatically sensing the time a user spends in a geolocation through the user's smart device, and the average service wait time is calculated based on manual user feedback provided by users after the service is rendered. For an entertainment service dealing with space renting, a user may view that the most critical metric is the maximum time allowable to reserve an allocated space. For a Spa service, a metric of interest could be the level of noise in the background and availability of background music (e.g., type and genre). Hence, it is important to rank SPs based on performance metrics specified by the user.

Existing service ranking systems have several drawbacks. First, current service ranking systems lack flexibility to allow a user to specify its own performance metrics to rank SPs, so very frequently a user would need to read long and inaccurate reviews to discover if an SP satisfies his/her need. Second, current service ranking systems depend highly on manual user feedback, crowdsourcing, and shared location-based information to obtain knowledge about SPs.

The associate editor coordinating the review of this manuscript and approving it for publication was Agustian Taufiq Asyhari.

There is no good mechanism to cope with ballot-stuffing attacks or bad-mouthing attacks. Third, there is virtually no design allowing a ranking system to be scalable with a huge number of IoT devices that can provide feedback concurrently. Fourth, all current service-ranking systems are essentially based on reputation rating [5], offering references on whether an SP is good or bad based on the majority-based review ratings. There is no mechanism that allows a user to filter recommendations based on the similarity in taste/interest or social relationships with the reviewers, so the user can filter out reviews reported by other users who have dissimilar taste/interest or no social relationship.

The above issues are addressed in this paper by proposing the notion of "*smart service community*," namely SSC. For scalability, we propose that an SSC be built as a cloud utility allowing users (i.e., owners of IoT SPs and SRs) to register for the service community. The SSC provides service rating reports after a service is rendered and makes credibility reports available based on recommendations received. Above all, the SSC allows an SR to select an SP for a requested service and query the system for ranking the SPs based on the *service quality performance metrics* specified by the SR. Due to the services provided by the SSC, a user would simply access the cloud utility via a mobile application installed in his/her IoT devices, such as smart phones.

To overcome the drawbacks of reputation-based service ranking systems, we propose *subjective-trust-based service management*. That is, for each performance metric specified by an SR, the cloud utility maintains an "one-to-one" subjective SR-SP trust score toward an SP for the SR as well as an "one-to-one" subjective SR-SR credibility trust score toward another SR who serves as a recommender (which we call a witness in this paper). Unlike a reputation system [5] based on "common beliefs" of all evaluators, the key design of our trust-based SSC management lies in the notion of "subjective" trust with one-to-one trust evaluation in both the SR-SP service trust score and the SR-SR credibility trust score. In particular, an SR-SR credibility rating report submitted from an SR to the cloud utility reflects the extent to which whether the service rating recommendation from another SR (serving as a witness [2] or a recommender) toward an SP is similar to that of the SR itself. A high SR-SR trust score means a high "taste similarity" between the SR and a witness SR. If the credibility score of a witness SR is high, the witness's recommendation is integrated into the overall SR-SP trust score computation; otherwise, it is discarded or treated with a small weight. Our notion of *subjective, one-to-one trust* allows an SR to filter out recommendations reported by SRs who have dissimilar taste/interest in rating a service from the same SP. Even if the majority of SRs perform ballot-stuffing or bad-mouthing attacks on an SP, an SR would accept only recommendations from SRs with high "taste similarity" for computing the SR-SP trust score.

The new design notion of SCC is exemplified with a smart food service community for which the major performance metric is the wait time for food service. The example SCC under our trust-based service management design is demonstrated to outperform contemporary service ranking systems.

The key contributions of this paper are as follows:

1. In this paper the notion of a "smart service community" (SSC) is proposed to address the grand challenge of a huge number of IoT devices providing similar services in a smart city environment. We are the first to suggest that an SCC be built as a cloud utility accessible via a mobile application installed in user-owned IoT devices, such as smart phones and exemplify how such cloud utility can be built for a smart food service community in a smart city setting.
2. To the best of our knowledge, we are the first to leverage smart IoT sensing technology for a user to automatically rate a provider's service and rate the feedback from provider evaluators in an SSC. In other words, an SR can generate SR-SP service ratings as well as SR-SR credibility ratings through sensing on sound, lighting, smell, geolocation, spaciousness of premises, WiFi duration, and availability, all through smart IoT devices.
3. Novel trust-based service management techniques are developed specifically for SSCs. Unlike existing service rating systems using reputation [5] (i.e., a common belief by the majority of evaluators) for rating SPs, we use the concept of "subjective" one-to-one trust relationship; hence, both the ratings toward service providers and the credibility of SR witnesses are subjective and one-to-one (based on own belief), reflecting the dynamic relationships between each pair of nodes in the system. An analytical formulation for calculating a subjective trust score and witness credibility score is provided, where an analytical formulation for calculating the subjective trust score and witness credibility score is provided by Josang's Beta Reputation System being utilized while accounting for time-decayed positive and negative experiences. Our design, through detailed performance evaluation, is proved to be highly resilient to self-promotional attacks (by malicious SPs) and false recommendation attacks (by malicious witnesses) [6], [7].
4. New credibility assessment techniques are developed, including "taste similarity credibility," "participation credibility," and "location credibility" techniques to assess whether a performance metric reported by a witness (e.g., service wait time for food service) is trustworthy. The witness credibility is then used to filter recommendations received from witnesses to compute the overall trust score of an SP for decision making. Our protocol resilience against a high percentage of malicious SRs is demonstrated and compared with baseline service rating systems.

The rest of the paper is organized as follows. Section II surveys the related work. Section III discusses the assumptions and system model, including the threat model. Section IV describes our trust management protocol for trust-based service management of SSCs in detail. Section V conducts a simulation study for a smart food service community to evaluate the merit of our approach, and performs a comparative analysis with two baseline service ranking systems. Finally, Section VI concludes the paper and suggests future work directions.

II. RELATED WORK

Trust is an effective mechanism for achieving trustworthy service. Direct experiences can provide a customer accurate information about an SP because personal experiences are most trustworthy. Josang's Beta Reputation System [5] is a well-known protocol for assessing "direct trust." The basic idea is to take binary ratings as input (i.e., positive or negative experience) and compute an SP's trust score by statistically updating the Beta distribution probability density function such that the posteriori (i.e., updated) trust score is computed by combining the priori (i.e., previous) trust score with new evidence observed. Our work also adopts Josang's Beta Reputation System [5] for an SR to assess the service rating of an SP based on direct experiences.

If an SR has never had any prior service experiences with an SP, then recommendations would be needed. Identifying trustworthy recommendations is challenging. In the literature, various recommendation filtering methods have been developed to filter untrustworthy recommendations. One method is social similarity based "collaborative filtering" by which a recommendation is considered trustworthy when the user providing the recommendation has a high degree of social similarity with the SR because a high social similarity between the SR and a witness implies trustworthiness. Conversely, a recommendation is considered untrustworthy when the witness providing the recommendation has a high degree of social similarity with the SP [8], [9] because a high social similarity between the SP and a witness implies collusion. Nitti *et al.* [10] described how one can build a friendship social graph to rate a recommender node based on the friendship between the recommender and the SR or SP. Another method is based on the concept of belief discounting [5] by which a discount is applied to a recommendation based on the amount of trust an SR has toward the recommender. That is, when A receives a recommendation from B about C on a service item, the recommendation will be "discounted" based on the degree to which A trusts B on the service item.

Relative to [5], [8], and [9], we develop "taste similarity credibility," "participation credibility," and "location credibility" design concepts to assess the overall credibility of a witness toward a user (See Section IV.B.2 for details). In our SSC model, the credibility of a recommender represents the extent to which the recommender is trustworthy because the recommender and the user are similar in taste in ranking a service. Therefore, when an SP is being evaluated on a specified user performance metric (e.g., service quality), the recommender and the user would provide a similar service rating. We propose to use taste-similarity based credibility to filter recommendations received from witnesses so that untrustworthy recommendations (subjectively from the user's perspective) will be filtered out for computing the user's overall trust score toward an SP that is being evaluated on a specific user performance metric. Furthermore, unlike a reputation-based system [5], our trust system is one-to-one and subjective by which trust evidence can be collected and integrated such that each SR can do one-to-one subjective trust assessment toward each SP and one-to-one subjective credibility assessment toward each witness. As a result, an SR accepts only recommendations from witnesses with high credibility scores for computing the overall trust score of an SP, thereby effectively fending off recommendation attacks (i.e., bad-mouthing and ballot-stuffing attacks) even if the majority recommenders are malicious.

In the trust management domain for IoT systems [11], Chen *et al.* [8], [9] used social similarity including friendship, community of interest, and social contact relationships to rate recommenders or witnesses assuming that a recommender having a close social relationship with an SR would tell the truth. Unlike [8] and [9], our trust system does not explicitly maintain social relationships between any two IoT devices for assessing credibility since it may be difficult to obtain such social information due to privacy reasons. Rather, we use a "service rating similarity" of two SRs toward the same SP at the same time after a service is rendered by the SP in lieu of the social similarity. The reason is that if two SRs give a similar service rating toward an SP's service rendered in the same time frame, then a high taste similarity between the two SRs is expected, implying that their view toward a particular performance metric of the SP is about the same. Nitti *et al.* [12] proposed a centralized IoT trust management system called ObjectiveTrust that assesses the trust score of an IoT device node through a weighted sum of the "centrality" score and the average opinion score in both long term and short term after applying the recommender's credibility score toward the SP to filter untrustworthy recommendations. Their credibility score is also based on social similarity. However, they estimated the credibility score of a recommender by assuming that a recommender having a close social relationship with an SP would likely collude with the SP and lie, and consequently be assessed with a low credibility score. Their work assumes the presence of a social network graph that can reveal the social relationships between any pair of nodes in the system. In addition to privacy concerns, scalability is another concern for dynamically maintaining an accurate social network graph for a large-scale IoT environment, which is very often not available in practice. Furthermore, ObjectiveTrust computes the "objective trust" (i.e., a common belief or reputation), not the "subjective trust" of an IoT device, which is not feasible given that such a social graph is very likely unavailable. Our work differs from [12] in that we do not assume the presence of a social network graph and we consider "subjective" trust, including both the one-to-one SR-SP trust score and SR-SR credibility score to effectively fend off recommendation and collusive attacks.

Many applications are based on collected user crowd-sourced geo-referenced data where trust management is used to identify trustworthy data for decision making [13]. Prandi *et al.* [14], [15] proposed a trust-based system which collects user mobile data regarding points of interest to map urban accessibility, which in turn provides users with disabilities personalized paths based on their preferences and needs. They assumed that data sources include regular users (and their sensors) and trustworthy experts (e.g., local authorities, associations). Through mobile device sensors, users can identify barriers and facilities for reporting. Prandi et al. [14], [15] additionally relied on a gold set [16], [17] obtained from trustworthy experts. A user's trust is updated by relying on comparisons with the majority of users when expert data is unavailable. Unlike an accessibility community, service providers have competing interests where both service providers and recommenders could be tempted to supply false recommendations to increase their interests, with no clear authoritative source of information to rely on. Vidya and Nandini [18] proposed a trust-based protocol for a smartphone application with friendship in social media being the main factor characterizing behavior. The authors used both direct and indirect observations to build trust, where direct trust is based on prior friendship behavior and community membership, and indirect trust is based on recommendations. Our system also relies on users to collect geo-referenced data; however, we consider the use of IoT-assisted technology to collect measurements of user-specified metrics. Furthermore, the above works considered only objective trust and did not consider recommendation attacks.

Lin and Dong [19] proposed a social IoT model with the aim of clarifying trust concepts in social IoT. The authors identified limitations of current social trust IoT protocols, such as dependence on single evaluation factors/metrics, unilateral evaluation of trustor to trustee, consideration of only the success rate of the main task, and lack of consideration of dynamic environments. However, their work fails to consider temporal and spatial factors which can be utilized (e.g., for assessing the quality of an SP and a witness). Nitti *et al.* [20] discussed the importance of feedback in trustworthiness management in the IoT. They stressed the importance of both short-term context associated with environmental conditions and long-term context, including the trustee Quality of Experience, for accurate feedback assessment. Relative to the works cited above [19], [20], we consider one-to-one subjective trust assessment for both SR-SP service rating and SR-SR credibility rating assessments; in addition, we leverage IoT-assisted technology for collecting measurements of user-specified performance metrics, taking into account various context factors related to time, location, and capability of IoT devices.

Xia *et al.* [21] proposed a light-weight subjective trust inference framework based on both trust assessment and prediction for MANETs. The subjectivity comes from each node's direct trust assessment toward a node's behavior with regards to packet reception for predicting routing paths with minimum overhead. Wu *et al.* [22] and Su *et al.* [23] considered collaborative filtering with social similarity for service recommendations. Abderrahim *et al.* [24] proposed a centralized trust management system aiming to find the most trustworthy service provider for social IoT. They developed a trust module for trust and reputation computing, and a learning module for behavior classification and decision making. Ding *et al.* [25] considered a time-aware service recommendation approach using similarity enhanced collaborative filtering. The authors used an autoregressive integrated moving average (ARIMA) method to predict future QoS values. After considering different QoS indicators, the system finally recommends the top k candidate services. However, the above cited works [21]–[25] did not consider one-to-one subjective evaluation, thereby introducing high susceptibility to recommendation attacks. Also, there was no consideration given to the characteristics of IoT in rating members and verifying recommendations.

The notion of "smart service community (SSC)" is coined in this paper, which is novel. In the literature, we found only the parking community proposed by Timpner *et al.* [3] is close to our notion of the SSC. However, there are no SPs in their parking community since parking spaces in [3] are not parking facilities actively competing for parking services, which would otherwise act as SPs in an SSC considered in our paper. Their parking community merely comprises SRs whose trust scores are updated based on real-time information to check if they (as recommenders) tell the truth about whether a parking space is free. Our notion of the SSC differs from the parking community in [3] in that a service community comprises of both SPs and SRs with SPs actively competing for providing services that may be interesting to members of the service community. We use real-time information to assess not only the trustworthiness of an SP, but also the credibility of an SR (as a recommender). Moreover, the SR-SP trust score or SR-SR credibility score evaluation is "subjective" rather than "objective" in order to mitigate recommendation and collusive attacks.

III. SYSTEM MODEL AND ASSUMPTIONS

A. SYSTEM MODEL

Our proposed trust-based service community management system is based on a centralized cloud utility in the service community [26] collecting trust evidence from SRs. Ideally, it can run as part of a service ranking application (e.g., Yelp) with mechanisms in place to collect positive/negative service experiences from SRs and rate SPs. Our notion of an SSC is geared toward a specific type of service system typically in a smart city setting, e.g., a city parking service system, an Italian food service system, a walk-in clinic service system, a bubble teashop service system, a phone repair system, a Spa service system, a children daycare system, a car wash system, etc. A customer seeking for each type of service would access the corresponding cloud utility that provides service rankings of SPs in the SSC. Very frequently there is a specific set of "service quality performance metrics" that

TABLE 1. Service Quality performance metrics for example smart service communities (SSC).

	Service quality performance metric													
Smart Service Community Type	Service wait time	Parking accessibility	Waiter : table ratio	Time allowed after service	Payment options	Product variety	Size/indoor spaciousness	Temperature at premises	Lighting	Noise	Music played (genre, type)	Cellular coverage	Wifi availability	Wifi duration
Dine in restaurant	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Walk-in clinic	✓	✓	-	-	✓	-	✓	✓	-	✓	-	✓	✓	✓
Phone repair shop	✓	✓	-	-	✓	✓	-	✓	-	-	-	-	-	-
Spa/salon	✓	✓	-	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Children's daycare	-	✓	-	✓	✓	-	✓	✓	✓	✓	-	✓	✓	✓
Car wash	✓	-	-	-	✓	✓	-	-	-	-	-	-	-	-

are of interest to the SRs in the SSC. Table 1 lists possible "service-quality performance metrics" for some example SSCs to which our trust-based service management design is applicable.

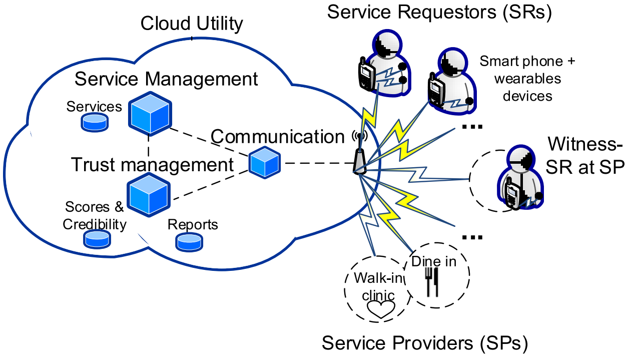


FIGURE 1. System design space of the IoT trust-based Smart Service Community management system.

As shown in Fig. 1, an SSC consists of three main actors interacting with one another as follows:

1. *Cloud Utility* (CU): The CU is a central storage and processing unit of the system. The CU allows SPs and SRs to register in the system so that they can be authenticated. The CU manages service ratings and trust scores and provides service portals to allow SPs to advertise one or more services offered by them. The CU also provides service portals for individual SRs to query and rank SPs based on service quality performance metrics selected by SRs. The CU is a trusted entity, such as Yelp, in a smart food service community.
2. *Service Providers* (SPs): An SP must register with the CU, so it can advertise its services through the CU. Every registered SP must post its performance data for service quality performance metrics that would be of interest to SRs in the service community. The posting frequency may depend on the nature of the performance metric, e.g., for the "service wait time" performance metric, it can be posted on a minute-by-minute basis. However, an SP can be malicious and therefore a trust system is needed to rate the trustworthiness of the SP.
3. *Service Requesters* (SRs): Every customer who wants to use services advertised in an SSC is an SR. An SR can be equipped with IoT-assisted technology, i.e., an SR can carry handheld or body wearable IoT devices with sensory and communication capabilities to automatically collect and report measurements of certain performance metrics related to a service. Therefore, an SR can report information in two ways:
   - *Manual:* An SR is prompted by the SSC application running on the SR's smart IoT device (e.g., a smartphone) to provide feedback consisting of answers to CU-prepared questions regarding specific service quality performance metrics of an SP. This is the default method of collecting metric measurements when IoT sensing technology is unable to measure certain user-specified performance metric (e.g., payment options).
   - *IoT-assisted:* An SR uses its IoT device equipped with smart sensing technology to automatically collect SPs' service performance measurements. This in effect creates the feedback automatically to be sent to the CU. Using sensors can provide more accurate readings for certain types of service quality performance metrics (e.g., 21 °C room temperature at a restaurant) which otherwise could be misinterpreted. IoT sensing technology can measure ambient environment changes, including lighting, noise, flavor, smell, background music detection, cellular coverage, and WiFi availability, which in turn allows certain metrics of interest to be automatically measured. In a smart food service community, the total service time of an SR at an

SP location may be measured by a sudden change of the ambient environment in lighting, noise, and background music. The service wait time may be measured by sudden changes in flavor and smell while all others remain the same.

An SR can send a query to the CU to rank SPs based on a set of performance metrics selected by the SR. An SR can also serve as a witness to help other members of the SSC to find the best SP based on their own set of service quality performance metrics. However, since a witness can be malicious, a witness rating system is needed to rate the credibility of each witness. Specifically, an SR can send an "SR-SP service rating" report to the CU after personally receiving a service from an SP. An SR can also send an "SR-SR credibility rating" report toward a witness.

TABLE 2. Notations.

Parameter	Meaning
$M_{adv}^{SP_j,m}$	SP_j's advertised performance value for metric m
$ST_{SR_i}^{SP_j}$	$SR_i's$ subjective trust score toward SP_j
$M_{SR_i}^{SP_j,m}$	SP_j's projected performance value for metric m
$M_{adv}^{SP_j,wait\_t}$	SP_j's advertised service wait time
$M_{SR_i}^{SP_j,wait\_t}$	SP_j's projected service wait time
$T_{SR_i}^{SP_j}$	Service rating of SR_i toward SP_j
$CR_{SR_i}^{SR_j}$	Credibility score from SR_i toward SR_j
CR_{th}	Credibility threshold
ST_{th}	Subjective trust threshold
$Seen_{i,k}^{t}$	Set of all SRs seen by SR_i at location SP_k at time t
$AS_{X,k,j}^{t}$	Set of all SRs who saw SR_j at location SP_k at time t
$ST_{SR_i}^{SP_j,all}$	$SR_i's$ subjective trust score toward SP_j across all metrics

Table 2 lists the notation used in our protocol design. A querying SR, say SR_i, first searches for a set of qualified SPs based on a set of "service quality performance metrics" deemed as important by the SR. These user-specified service quality performance metrics are formalized into search criteria and are executed against the CU database through the service portals. The CU then returns a set of SPs that satisfy the criteria. Each qualified SP returned, say SP_j is associated with the following information for SR_i's decision making: (1) $SR_i's$ subjective trust score toward SP_j denoted by $ST_{SR_i}^{SP_j}$;(2) $SP_j's$ advertised performance for metric m denoted by $M_{adv}^{SP_j,m}$; (3) $SP_j's$ projected performance for metric m denoted by $M_{SR_i}^{SP_j,m}$; and (4) a list of witnesses who had service experience with SP_j along with their reported service ratings for metric m (on the scale of 1 to 5) and the actual SP_jperformance experienced for metric m.

For example, if we consider the performance metric of interest to $SR_i$is the "service wait time" (i.e., time to wait until service is rendered), SR_i can send a query to the CU with search criteria including location, cuisine type, and service wait time (assuming it is the only performance metric labeled as "*wait_time*" in this use scenario) to find SPs that satisfy the search criteria.

The CU would return a query result containing a record for each SP_j satisfying the search criteria. The record associated with SP_j would contain (1) $SR_i's$ subjective trust score toward $SP_j's$service wait time performance; (2) $SP_j's$ advertised service wait time $M_{adv}^{SP_j,wait\_t}$;(3) $SP_j's$ projected service wait time $M_{SR_i}^{SP_j,wait\_t}$; and (4) a set of witnesses having service experience with SP_j, along with the service ratings toward SP_j on the scale of 1-5 for metric "*wait_t*" and the actual wait time (in min.) experienced for "*wait_t*."

Using the information returned by the CU, a querying SR would select what it believes to be the best SP for the service it requested. After the service is rendered, the SR assesses its own service rating for a metric of interest (say metric *m*) based on actual experience. The SR serving as a witness would then report its service rating as well as the actual performance experienced for metric *m* to the CU through the service portal provided by the CU. The service rating and the actual performance experienced for metric *m* reported by an SR toward each SP will be stored in the CU such that when another SR runs a search query for which an SP is a match, the witness information associated with the SP will be included in response to the querying SR. The stored witness information used for the response is organized based on the service quality performance metric type and granularity since some metrics are relatively static (e.g., payment options), while others may be highly dynamic depending on the time of day, and day of week (e.g., service wait time).

B. THREAT MODEL

We consider malicious SPs and SRs (when acting as witnesses) as the only form of attack. Other forms of attack that are possible in a service community, such as Distributed Denial of Service (DDoS), system intrusion, and data leaks [27], are beyond the scope of this paper. Our threat model covers the following attacks:

1. *Self-promoting attack:* A malicious SP can advertise a false performance level for a particular metric to attract more customers. For example, an SP would advertise a shorter service wait time than it actually is and a customer who selects it for service would experience a much longer wait time.
2. *Recommendation attack:* A recommendation attack occurs when a malicious SR (as a witness) colludes

with other malicious SPs to increase their chance of being selected for service. Two forms of the recommendation attack include:

- *Ballot-stuffing attack:* In order to gain profit(s) by attracting more customers, a malicious SP can advertise a false performance level and other malicious SRs may backup the false performance. An SR is more likely to trust an SP if many SRs acting as witnesses support the SP's advertised performance level.
- *Bad-mouthing attack:* Malicious SRs acting as witnesses may report a false performance level for a good SP (e.g., a large wait time) in order to ruin the reputation of the good SP. An SR is more likely to distrust an SP if many SRs refute the SP's advertised performance level.

3. *Non-cooperation attack:*Malicious SRs acting as witnesses may exhibit selfish behavior by utilizing the CU's resources for search functions but are unwilling to produce reports to the CU.
4. *Report fabrication attack:* Reports may be fabricated by malicious SRs acting as witnesses by providing false locations [6], [7], pretending to be at an SP's location at time t to rate the SP when in fact they are physically elsewhere.

In the following section, we explain how our trust management protocol design copes with these attacks.

IV. TRUST MANAGEMENT PROTOCOL DESIGN

In this section, we describe our trust protocol design in detail. We first describe our design principles in Section IV.A. Then in Section IV.B, we describe the computational procedure for computing the one-to-one SR-SP subjective trust score for a user service quality performance metric specified by the SR. Since the computational procedure is the same for every user service quality performance metric specified by the SR, we shall generally refer a selected metric as "metric m" and omit saying "for metric m" when the context is clear Section IV.B is organized as follows: (1) Section IV.B.1 describes how an SR rates an SP based on own experiences. (2) Section IV.B.2 details how an SR rates a witness. These two pieces of rating information are reported to the CU who keeps track of pair-wise SR-SP and SR-SR trust scores. It is "subjective" because the computation is based on the subjective view of the SR who issues a query to the CU (about which SPs and witnesses are trustworthy based on own experiences). (3) Section IV.B.3 explains how the CU computes an SR's "subjective" trust score toward an SP. (4) Section IV.B.4 discusses how the CU computes an SP's projected performance (for metric m). The SP's projected performance conveys the system's belief based on all collected service ratings and trust evidence. This projected performance provides an expected service quality if this particular SP is selected for service. (5) Section IV.B.5 discusses how to select the "best" SP for service among all qualified SPs as application-level decisions. Finally, Section IV.C describes decision making strategies by an SR when there are multiple service quality performance metrics.

A. DESIGN PRINCIPLES

The following design principles are adopted in our SSC trust management protocol design:

1. *Own experiences outweighing witness experiences:* When an SRissues a query to the CU asking for the trust score and the projected performance of an SP, the calculation should heavily count on this particular SR's own experience (if any), especially in an environment with a high concentration of malicious witnesses.
2. *Subjective trust computation*: An SR judges whether an SP or a witness is good or bad based on its subjective view, rather than based on the "common belief" of all SRs (e.g., objective trust). The subjective view is exercised when an SR evaluates an SP based on own service experience as well as when an SR selects witnesses based on its own belief that these witnesses are highly credible and can provide true witness experiences about an SP.
3. *Witness filtering based on witness credibility*: Each SR has its own one-to-one SR-SR credibility score toward a witness because trust (or credibility) rating is subjective and one-to-one. Witnesses with low credibility will be filtered out to deal with recommendation attacks.
4. *Scalability*: For scalability, the one-to-one SR-SP trust information and one-to-one SR-SR credibility information are not to be kept in each SR. Rather, the CU in the cloud maintains all trust and credibility information and performs all necessary computation to answer a query issued by an SR regarding the SR-SP trust score and an SP's projected performance.

B. ONE-TO-ONE SR-SP SUBJECTIVE TRUST SCORE FOR A PERFORMANCE METRIC SPECIFIED BY A USER

1) SR-SP SERVICE RATING SCORE BASED ON OWN EXPERIENCE

When an SR receives service from an SP, it will compare the SP's advertised performance (of a service quality metric, such as wait time) with the SP's actual performance observed. We consider the service rating ranged from 1 to 5 as an integer, with 1 being least satisfaction and 5 being most satisfaction. Specifically, if the actual performance value is less than 20% over an SP's advertised performance value, then the service rating is 5, 20-39% then 4, 40-59% then 3, 60-79% then 2, and 80%-100% then 1.

Josang's Beta reputation system [5] is utilized to calculate the SR's service rating toward an SP based on the historical service transactions between the two entities. An SR i, denoted by SR_i, assesses the service rating of an SP j, denoted by SP_j, by:

$$T_{SR_i}^{SP_j} = \frac{\alpha}{\alpha + \beta} \tag{1}$$

$$\alpha = \sum_{all} f^{SP_j}_{SR_i}(t) e^{-\lambda_d (t_{now}-t)} \tag{2}$$

$$\beta = \sum_{all} (1 - f^{SP_j}_{SR_i}(t)) e^{-\lambda_d (t_{now}-t)} \tag{3}$$

where α and β are the cumulative amounts of positive and negative experiences, respectively, with time decay from SR_i toward $SP_j$with $\alpha = 1$ and $\beta = 1$ initially given. Here $f^{SP_j}_{SR_i}(t)$ represents the amount of positive experience of the rating computed at time t and can be computed as $f^{SP_j}_{SR_i}(t) =$ (service rating at time t)/5, resulting in a service satisfaction level that is either 1, 0.8, 0.6, 0.4, or 0.2. The term $e^{-\lambda_d (t_{now}-t)}$ in (2) and (3) represents time decay where $t_{now}$is the current time, and λ_d is the decay parameter to discount old service experiences. By using time decay, more recent experiences by SR_i toward SP_j are given a higher weight toward $T^{SP_j}_{SR_i}$. The service rating score is updated whenever SR_i receives service from SP_j and subsequently assesses whether it is a positive or negative service experience. To support the design principle of "scalability" each SR reports its service rating toward an SP to the CU who maintains $T^{SP_j}_{SR_i}$ for all pairs of SR_i and SP_j. The process of assessing an SP's service rating is depicted in Fig. 2.

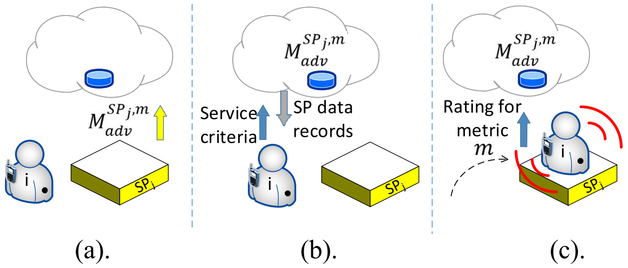


FIGURE 2. The process of assessing SP_j 's trust score. (a) SP_j advertises its performance for metric m to CU, (b) SR_i sends a query to the CU containing the performance metrics required. The CU finds all SPs that satisfy the query including SP_j, and (c) SR_i chooses SP_j based on application-level decision making, visits SP_j to obtain the service, and then compares SP_j advertised performance for metric m with the actual performance experienced to obtain a new service rating for metric m (using smart IoT devices) and reports the service rating for m to CU.

2) SR-SR CREDIBILITY SCORE

The trustworthiness of a witness (who is an SR acting as a witness) is rated by its credibility. The credibility score of a witness is based on (1) the "taste similarity" in reporting similar SR-SP service ratings for a performance metric (called taste similarity credibility or *tsc*), (2) the participation in reporting to the service community via CU (called participation credibility or *pc*), and (3) the location credibility of reports (called location credibility or *lc*), all from the perspective of the "trustor" SR toward a "trustee" SR, thus resulting in an one-to-one SR-SR credibility score. An observed/logged event relating to these factors can be represented by a level of satisfaction from the trustor SR toward the trustee SR, which in turn is sent to the CU to update the pair-wise SR-SR credibility score. We denote the credibility score from SR_i toward SR_j by $CR^{SR_j}_{SR_i}$, given as:

$$CR^{SR_j}_{SR_i} = \frac{a}{a+b} \tag{4}$$

$$a = \sum\nolimits_i w_i \times a_i \tag{5}$$

$$b = \sum\nolimits_i w_i \times b_i \tag{6}$$

where a_i and b_i represent the aggregate levels of satisfaction and dissatisfaction, respectively, for credibility type i. w_i represents the allocated credibility weight given by the SR for credibility type i where $i \in \{tsc, pc, lc\}$ discussed above and the chosen weights are restricted by $\sum_i w_i = 1$. The weights, w_{tsc}, w_{pc}, and w_{lc}, should be adjusted to reflect the importance of various credibility types in an SCC. For example, participation may not be mandatory by all witnesses for a particular application and in this case w_{pc} would be low. Next, we discuss each of the credibility types in more detail.

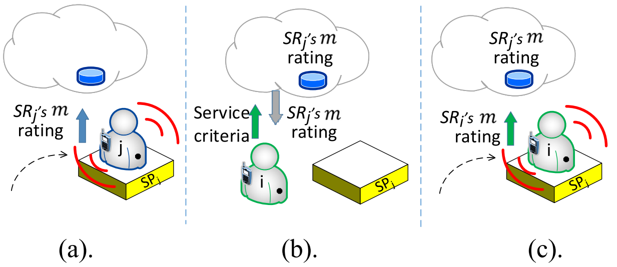
a: TASTE SIMILARITY CREDIBILITY

A witness reports its service rating toward an SP for a performance metric of interest after service is rendered. Once an SR itself experiences service from the same SP, it will compare its own service rating with the service rating reported by a witness to assess the witness's taste similarity credibility. If the service rating is similar, then it means that they have a high level of "taste similarity" in rating the same SP with respect to the performance metric in question and the witness is considered trustworthy because the SR and the witness share a similar taste toward the same service provided by the same SP. A witness's taste similarity credibility rating is ranged from 1 to 5 as an integer, representing the difference between the SR's service rating and the witness's reported service rating. For example, if we consider the metric of interest is "wait time," then one way to obtain the taste similarity credibility rating is to find the absolute value of an SR's own wait time service rating minus the witness's wait time service rating denoted by d. Then the taste similarity credibility rating is $5 - d$. So if the two wait time service ratings are the same, then $d = 0$ and the taste similarity credibility rating is 5. If the SR's wait time rating is 5 but the witness's wait time rating is 1, then the taste similarity credibility rating is $5 - d = 5 - 4 = 1$. To support the design principle of "scalability," an SR reports the resulting taste similarity credibility rating about a witness to the CU who keeps a database of all pair-wise SR-SR credibility ratings. Specifically, the cumulative amounts of time-decayed positive and negative experiences from SR_i toward SR_jcan be derived as:

$$a_{tsc} = \sum_{all} g^{SR_{j,tsc}}_{SR_i}(t) \times d_{tsc}(t) \tag{7}$$

$$b_{tsc} = \sum_{all} (1 - g^{SR_{j,tsc}}_{SR_i}(t)) \times d_{tsc}(t) \tag{8}$$

with $a_{tsc} = 1$ *and* $b_{tsc} = 1$initially, and $d_{tsc}(t)$ is a decay factor for the taste similarity credibility rating where $d_{tsc}(t) = e^{-\lambda_d (stdev(t_{now}, t, LWR_j(t)))}$ Here a_{tsc} and b_{tsc} are accumulated over all witness assessments, each occurring at a separate time t when a witness report is received by the CU. $g_{SR_i}^{SR_j,tsc}(t)$ is the amount of positive experience of the taste similarity credibility rating at time t computed based on the last provided SR_j taste similarity credibility rating at time t, computed by $g_{SR_i}^{SR_j,tsc}$ = (witness taste similarity credibility rating at time t)/5, resulting in a service satisfaction level that is either 1, 0.8, 0.6, 0.4, or 0.2. The term $e^{-\lambda_d(stdev(t_{now},t,LWR_j(t)))}$ represents time decay where $\lambda_d$is the decay parameter and $stdev(t_{now}tLWR_j(t))$ is the standard deviation of the time of a current visit to SP_j (t_{now}), time of SR_i judging witness SR_j (t), and the result of function LWR_j (t), which is the time of the last witness report from witness SR_j rating SP_j prior to time point t. This keeps the calculated $CR_{SR_i}^{SR_j}$more relevant at the time of assessment by minimizing the weight of old data. In addition, this way allows fair assessment so that reported taste similarity credibility ratings by the witness cannot be expected to be similar to current taste similarity credibility ratings if there is a large time gap. Fig. 3 shows the process of computing the SR_i-SR_j taste similarity credibility rating.



(a). (b). (c).

FIGURE 3. **The process of computing the SR_i -SP_j taste similarity credibility rating for metric m. (a) SR_j sends its service rating for metric *m* after visiting SP_j to the CU at time < *t*, (b) At time ≥ *t*, SR_i sends a query containing the performance metrics required to the CU. The CU finds all SPs (including SP_j) that satisfy the query and all witnesses' service ratings (including SR_j 's service rating for metric *m* toward SP_j), and (c) SR_i chooses SP_j based on application-level decision making, visits SP_j to obtain the service, and then compares its service rating for metric *m* with that of witness SR_j 's service rating for metric *m* (using smart IoT devices), and reports the SR_i -SP_j taste similarity credibility rating for metric *m* to the CU.**

b: PARTICIPATION CREDIBILITY

This credibility type determines if a witness exhibits selfish behavior and is utilizing the CU for search functions but is unwilling to produce reports to the CU, thus exhibiting the signs of a non-cooperation attack on the system. An SR could question the integrity of a witness if it detects non-cooperation. Let $Seen_{i,k}^t$be the set containing all SRs seen by SR_i at location SP_k at time t. If $SR_j \in$ ô$Seen_{i,k}^t$but is not included in the witness list contained in the query reply $QR_{i,k}^t$ sent by the CU, then SR_j did not review SP_k yet it was physically present at SP_k. This cross-checking can be done at a certain timepoint (e.g., the end of a day) where the seen lists for all visited locations by SR_i can be cross-checked against logged query replies by the CU. Alternatively, SR_i can take a more proactive approach by verifying on the spot by purposely sending a query to the CU at time t for SP_k, then examine the query reply of the CU to see if SR_j is in the witness list but $SR_j \in Seen_{i,k}^t$ locally on SR_i.

Furthermore, this instance of selfish behavior can be captured automatically by IoT-assisted technology. Specifically, SR_i's smart devices can confirm that SR_j is in fact at the SP_k by short-range communication (functioning as a substitute for physical eye sight). Such communication would be enabled a priori via an SSC app running on smart devices. SR_is device would then communicate directly (on behalf of its owner) with the CU to update SR_j's participation credibility accordingly without any mandatory manual entry. Alternatively, in the absence of IoT-assisted automatic verification, SR_i can rely on physical eyesight alone to first identify possible selfish witnesses and then send a query manually to the CU (e.g., using a smart phone) to initiate the verification as mentioned earlier. We model participation credibility as:

$$a_{pc} = \sum_{all} g_{SR_i}^{SR_j,pc}(t) \times d_{pc}(t) \tag{9}$$

$$b_{pc} = \sum_{all} (1 - g_{SR_i}^{SR_j,pc}(t)) \times d_{pc}(t) \tag{10}$$

with $\alpha_p c = 1$ and b_{pc}= 1initially, and $d_{pc}(t)$ is a decay factor for participation credibility where $d_{pc}(t) = e^{-\lambda_d(t_{now}-t))}$.

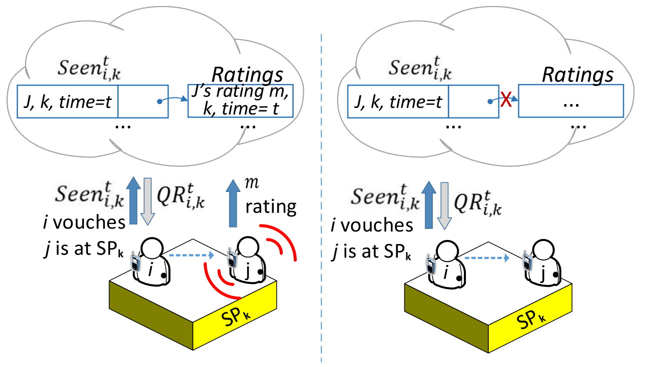
The purpose of the decay factor is again to give recent events of participation a higher weight. The positive experience (see (9)) of participation credibility for SR_j seen at location SP_k at time t is captured by $g_{SR_i}^{SR_j,pc}(t)$ as follows:

$$g_{SR_i}^{SR_j,pc}(t) = \begin{cases} C_{SR_i}, & SR_j \in Seen_{i,k}^t \wedge SR_j \in QR_{i,k}^t \\ 1 - C_{SR_i}, & SR_j \in Seen_{i,k}^t \wedge SR_j \notin QR_{i,k}^t \end{cases} \tag{11}$$

where C_{SR_i} is the capability of SR_i to identify other SRs within its vicinity. We set C_{SR_i}= 1 for using both IoT-assisted and manual verification, C_{SR_i}= 0.9 for IoT-assisted verification only, and C_{SR_i}= 0.8 for manual verification only. Thus, a selfish non-cooperative SR_j, as seen by SR_i will decrease its credibility score $CR_{SR_i}^{SR_j}$ thus decreasing the chance of SR_j influencing the subjective trust toward other SPs. The behavior can be further verified by application-level messages showing that SR_j was at the location and an interaction with the smartphone application was logged at that time. Equation (11) above models both the cases of positive participation evidence and lack of participation evidence as illustrated in Fig. 4 (a) and Fig. 4 (b), respectively.

c: LOCATION CREDIBILITY

The purpose of location credibility is to verify the correctness of an SR's location, thus identifying suspicious witness participation and witness SP report fabrication entailing a report fabrication attack. If SR_i is at location SP_k at time t and both $Seen_{i,k}^t$ and the query reply $QR_{i,k}^t$ show that SR_j



(a). (b).

FIGURE 4. Cases of SR_j 's participation credibility. (a) SR_j comes in contact with SR_j at location SP_k, adds SR_j to the seen list, then sends the seen list to the CU with $SR_j \in Seen^t_{i,k}$ and SR_j is matched successfully with the witness list for rating SP_k contained in the query reply $QR^t_{i,k}$, and (b) $SR_j \in Seen^t_{i,k}$ but there is no match with the witness list for rating SP_k contained in the query reply $QR^t_{i,k}$, indicating lack of participation.

was at location SP_k at time t, it marks positive results and sets $SR_j{}'s$ location credibility to C_{SR_i} the capability of SR_i to identify other SRs within its vicinity (Case 1). However, if SR_i is at location SP_k at time t and is unable to detect SR_j yet SR_j is in the witness list contained in $QR^t_{i,k}$, then SR_j might have fabricated this witness service rating and could be physically elsewhere. The CU can corroborate $SR_i{}'s$ suspicion regarding SR_j without invading $SR_j{}'$s privacy (avoid sharing actual location information at time t) as follows: SR_i first sends its seen list $Seen^t_{i,k}$ to the CU as a proof of $SR^{'}_j$s absence. The CU then cross-checks existing witness records for time t and identifies all SRs who have encountered j at location SP_k denoted by the set $AS^t_{X,k,j}$. If such SRs exist, then the witness in question was seen by the SRs and SR_i might have been unable to detect SR_j's presence at location SP_k at time t. The CU uses $SR^{\prime}_i$s saved credibility scores and checks if the aggregate credibility of the $AS^t_{X,k,j}$ set using credibility function CR, denoted by $CR(AS^t_{X,k,j})$, is over a desired credibility threshold CR_{th}. If yes, it marks positive results (Case 2); otherwise, it marks negative results (Case 3).

More specifically, we model location credibility as:

$$a_{pc} = \sum_{all} g^{SR_{j,lc}}_{SR_i}(t) \times d_{lc}(t) \tag{12}$$

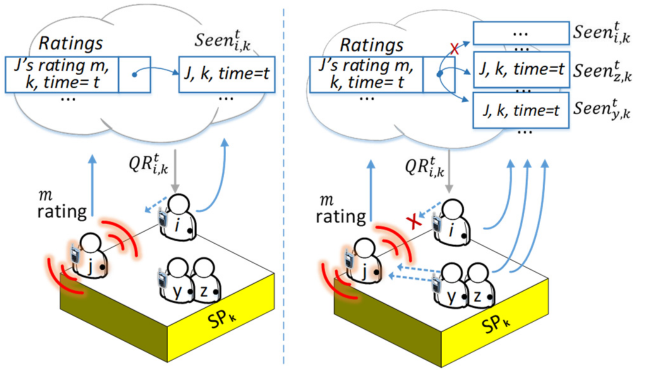
$$b_{pc} = \sum_{all} (1 - g^{SR_{j,lc}}_{SR_i}(t)) \times d_{lc}(t) \tag{13}$$

with $\alpha_{lC} = 1$and $b_{lc}= 1$initially, and $d_{lc}(t)$ is a decay factor for location credibility with $d_{lc}(t) = e^{-\lambda_d(t_{now}-t))}$. The purpose of the decay factor is to give recent events of location credibility a higher weight. The positive experience (because it is in (12)) of location credibility for SR_j listed in the query reply $QR^{k,t}_{SR_i}$ at time t is captured by $g^{SR_{j,lc}}_{SR_i}(t)$ as follows:

$$g^{SR_{j,lc}}_{SR_i}(t) = \begin{cases} C_{SR_i}, \left[\left(SR_j \in QR^{k,t}_{SR_i}\right) \wedge \left(SR_j \in Seen^t_{i,k}\right)\right] \\ \min\begin{pmatrix} C_{SR_i}, \\ CR(AS^t_{X,k,j}) \end{pmatrix}, \begin{bmatrix} \left(SR_j \in QR^{k,t}_{SR_i}\right) \\ \wedge\left(SR_j \notin Seen^t_{i,k}\right) \\ \wedge\left(\left|AS^t_{X,k,j}\right| > 0\right) \\ \wedge\left(CR(AS^t_{X,k,j}) \geq CR_{th}\right) \end{bmatrix} \\ 1 - C_{SR_i}, \begin{bmatrix} (SR_j \in QR^{k,t}_{SR_i}) \wedge (SR_j \notin Seen^t_{i,k}) \\ \wedge((|AS^t_{X,k,j}| > 0) \vee (CR(AS^t_{X,k,j}) < CR_{th})) \end{bmatrix} \end{cases} \tag{14}$$

$$CR(AS^t_{X,k,j}) = \frac{\sum_{x \in AS^t_{X,k,j}} CR^{SR_x}_{SR_i}}{\left|AS^t_{X,k,j}\right|} \tag{15}$$

where the first, second, and third cases in (14) match Case 1, Case 2, and Case 3, respectively. Fig. 5 (a) and Fig. 5 (b) illustrate Case 1 and Case 2 or 3, respectively. Note that all cases are based upon $SR_i{}'s$ subjective one-to-one view regarding other SRs (i.e., no common belief is used) and the CU only helps in determining who else has seen SR_j at location SP_k at time t.



(a). (b).

FIGURE 5. Cases of SR_j 's location credibility. (a) Case 1: SR_i is at location SP_k at time t and both $Seen^t_{i,k}$ and the query reply $QR^t_{i,k}$ show that SR_j was at location SP_k at time t, and (b) Case 2 or 3: SR_i is at location SP_k at time t and is unable to detect SR_j yet SR_j is in the witness list contained in $QR^t_{i,k}$. Therefore, SR_i asks the CU to cross-check the seen lists submitted by all SRs who have encountered SR_j at location SP_k at time t. Case 2 or Case 3 depends on whether $CR(AS^t_{X,k,j})$ is greater than CR_{th}.

3) SR-SP SUBJECTIVE TRUST SCORE

An SR's subjective trust score toward an SP has the physical meaning of whether the SR believes if the SP is malicious, i.e., lying about the service performance. The CU updates $SR_i's$ subjective trust score toward SP_jwhenever it receives a new $T_{SR_i}^{SP_j}$value (from (1)) or a new $CR_{SR_i}^{SR_j}$ value (from (4)). To support the design principle of "subjective trust computation instead of objective trust computation," the CU computes $SR_i's$ subjective trust score toward SP_j denoted by $ST_{SR_i}^{SP_j}$, as follows:

$$ST_{SR_i}^{SP_j} = \sum_{j=1}^{n_r} \left(\frac{CR_{SR_i}^{SR_j}}{\sum_{j=1}{}_r^n CR_{SR_i}^{SR_j}}\right) T_{SR_j}^{SP_j} \qquad (16)$$

where n_r is the number of witnesses having the highest credibility scores among all witnesses, $T_{SR_j}^{SP_j}$ (from (1)) is the service rating of SP_j as reported by SR_jacting as a witness, and $CR_{SR_i}^{SR_j}$(from (4)) is the credibility score of SR_j as evaluated by SR_i The idea behind (16) is that the lower the credibility of SR_j(acting as a witness), the lower the weight toward $ST_{SR_i}^{SP_j}$ computation. Thus, $ST_{SR_i}^{SP_j}$ is simply a weighted sum, i.e., the sum of the service rating scores reported from all SRs weighted by their respective credibility scores. To support the design principle of "own experiences outweighing witness experiences," we set the credibility of the querying SR (SR_i) to 1, i.e., $CR_{SR_i}^{SR_i}$= 1,to allow SR_i to be a first-hand witness (i.e., self-evaluation) and also to make $SR_i^{'}$s own experiences outweigh other witnesses' experiences. To support the design principle of "witness filtering based on witness credibility," we define a minimum credibility rating threshold (CR_{th})to filter out untrustworthy witnesses. Specifically, we only allow up to $n_r$witnesses whose credibility rating scores are higher than the minimum threshold. More specifically, to defend against bad-mouthing attacks (saying a good SR's credibility rating is low) and ballot-stuffing attacks (saying a malicious SR's credibility rating is high), witness filtering is applied by comparing $SR_i^{'}s$ own credibility rating toward $SR_j CR_{SR_i}^{SR_j}$ with $CR_{th.}$ If it does not pass the minimum threshold, SR_j is filtered out and its service rating recommendation $T_{SR_j}^{SP_j}$is discarded. If there is no witness SR qualified, then $n_r = 1$ in which case SR_i is the only witness (i.e., self-evaluation). If SR_i itself does not have any experience with SP_j, then there is no credible witness, in which case $n_r = 0$ and $ST_{SR_i}^{SP_j}$ remains the same as before.

4) SP'S PROJECTED PERFORMANCE

An SP, say SP_j advertises its performance for metric m (denoted by $M_{adv}^{SP_j,m}$) to attract customers to select it for service. However, SP_j may perform self-promotional attacks. To help a querying SR, say SR_i,understand if $SP_j's$ advertised performance data is false, as part of a query reply message, the CU returns the "SP_j's projected performance" (denoted by $M_{SR_i}^{SP_j,m}$) to SR_i based on a weighted sum calculation, as follows:

$$M_{SR_i}^{SP_j,m} = \sum_{j=1}^{n_r} \left(\frac{CR_{SR_i}^{SR_j}}{\sum_{j=1}{}_r^n CR_{SR_i}^{SR_j}}\right) M_{SR_j}^{SP_j,m} \qquad (17)$$

where $n_r$is the number of witnesses with the highest credibility selected among all (including self) that have reported service ratings of SP_j to the CU, and $M_{SR_j}^{SP_j,m}$is the actual performance value of SP_j as observed and reported to the CU by SR_j (acting as a witness). In (17), we assign a higher weight to a witness with a higher credibility, so the performance value reported by the witness with high credibility will dominate the resulting $SP_j's$ projected performance. In particular, the SR itself has a credibility score of 1, thereby ensuring that it (as a self-witness) has the highest weight among all witnesses.

5) APPLICATION-LEVEL USER DECISION MAKING

For each candidate SP_j that satisfies SR_i's service criteria and queries regarding service performance metric m, the CU returns a 4-tuple record to SR_i as follows: (a) $SR_i's$ subjective trust score toward SP_j ($ST_{SR_i}^{SP_j}$from (16)), (b) $SP_j's$ projected performance ($M_{SR_i}^{SP_j,m}$from (17)), (c) $SP_j's$ advertised performance ($M_{adv}^{SP_j,m}$) and (d) a list of all witnesses who have reported service ratings and performance values for metric m for SP_j. The decision to select the "best" SP for service among all qualified SPs depends on the application-level decision. For example, if the service quality performance metric is "wait time," the following three application-level user decision making policies can be considered (here metric m is "*wait_t*"):

1. *Select the least wait time SP*: That is, select the SP with the smallest $M_{SR_i}^{SP_j,wait\_t}$ value (i.e., the smallest wait time regardless of whether the SP is lying or not).
2. *Select the most trustworthy SP*: That is, select the SP with the largest $ST_{SR_i}^{SP_j}$ value (i.e., the most trustworthy SP).
3. *Select the least wait time SP among trustworthy SPs*: That is, select the SP with the smallest $M_{SR_i}^{SP_j,wait_t}$value among all qualified SPs whose $ST_{SR_i}^{SP_j}$value is no less than the minimum trust threshold ST_{th}.

Once SR_i selects the best SP, it proceeds to request the service from the best SP. After the service is rendered by the SP, SR_icompares its own service rating with each witness's service rating (available in the 4-tuple record returned by the CU) to update each witness's credibility rating to the CU.

C. APPLICATION-LEVEL USER DECISION MAKING WITH MULTIPLE SERVICE QUALITY PERFORMANCE METRICS

The CU applies the same computation procedure as described in Section IV.B for each user service quality performance metric selected by an SR. In case SR_i wants to apply more than one service quality performance metrics to evaluate SP_j,the CU will return to SR_i a 4-tuple record for each metric m_k as described in Section IV.B.5 earlier. How to make use of the information returned by the CU is a user-level decision. An SR, say SR_i,can evaluate an SP, say SP_j, by computing the overall subjective SR_i-SP_j trust score across all metrics, denoted by $ST_{SR_i}^{SP_j,all}$, as follows:

$$ST_{SR_i}^{SP_j,all} = \sum_{k=1}^{|M_{SP_j}|} w_{m_k} \times ST_{SR_i}^{SP_j,m_k} \tag{18}$$

where w_{m_k} is the weight given for metric m_k, $|M_{SP_j}|$ is the total number of metrics selected by $SR_i$to evaluate SP_j, and $ST_{SR_i}^{SP_j,m_k}$is the subjective SR_i-SP_j trust score for each metric m_k. The weight assignment is a user-level decision based on relative criticality/importance of all performance metrics selected by the user. Once $ST_{SR_i}^{SP_j,all}$is calculated, there are several selection strategies conceivable. For example, the user may want to select the most trustworthy SP across all metrics for service, i.e., the SP with the highest $ST_{SR_i}^{SP_j,all}$value is selected for service. Alternatively, a user may want to select the SP with the best projected performance value in one metric (e.g., least wait time) with the condition that the SP is reasonably trustworthy, i.e., the overall trust score $ST_{SR_i}^{SP_j,all}$ is above the minimum trust threshold ST_{th}.

V. EXPERIMENTAL EVALUATION

In this section, we conduct an experimental evaluation for a "smart Italian food service community" focusing on dine-in Italian restaurant SPs and using a single metric of "service wait time" (i.e., metric m is "*wait_t*") to exemplify our approach. We use an event-driven simulation tool called SMPL [28] for experimental evaluation.

TABLE 3. Parameter list for experimental evaluation.

Parameter	Meaning	Default	Range	Type
N_{SP}	Number of SPs	10	[5,20]	I
N_{SR}	Number of SRs	2200	[1000,3000]	I
λ_{SR}	Per-SR arrival rate	0.1/hr	[0.01-0.5] /hr	I
μ_{SP}	Per-SP service rate	1/hr	[0.5-3]/hr	I
m_{SP}	Per-SP service capacity	20	[10,30]	I
P_M	% of malicious SPs and SRs	30%	[10,50]%	I
R_f	Risk factor of a malicious SP	100%	[50,100]%	I
n_r	# of witnesses accepted	5	[3,10]	D
CR_{th}	Credibility threshold	0.6	[0.5,0.8]	D
ST_{th}	Subjective trust threshold	0.6	[0.5,0.8]	D
$t_{th,advertied}$	Advertised wait time threshold	30 min	[15,60] min	D
$t_{th,actual}$	Real wait time threshold	1 hr	[0.5,1.5]min	D
$ST_{SR_i}^{SP_j}$	SR_i's subjective trust score toward SP_j	0.5	[0,1]	O
$M_{SR_i}^{SP_j,wait\_t}$	projected wait time	-	-	O

Table 3 lists the parameters used in the experimental evaluation, including symbols, meanings, ranges, default values, and types. The column "Type" specifies the parameter type as *input*, *design* or *output*, denoted by I, D, or O, respectively. Input parameters serve to characterize the environmental and operational conditions. Design parameters characterize tunable system settings. Finally, output parameters realize the objective of the system. Only subjective trust scores and projected wait times are output parameters. Here we note that a test scenario in our experiment is defined by a set of input parameter values. The column "Range" specifies the range of each parameter. The column "Default" specifies the default parameter value used in our experimental evaluation. In the following paragraphs, we discuss the reasons why the default parameter values are chosen. In particular, the set of input parameter values are chosen such that it creates an environment in which there are sufficient malicious SPs as well as SRs (acting as witnesses), and the aggregate SR arrival rate is higher than the aggregate SP rate so that a malicious SP will have to cheat to lure SRs to use its service and an SR will have to carefully select the best SP for service based on its own specified performance metrics. We intentionally create the set of default input parameter values as listed in Table 3 to stress the system and it represents a design point of interest, because neither more demand than supply nor more supply than demand scenarios relative to our chosen test scenario would be interesting.

The parameter values are selected such that the aggregate SR arrival rate ($N_{SR}\lambda_{SR}$)is higher than the aggregate SP service rate ($N_{SP}\mu_{SP}m_{SP}$)by 50% so many SRs are forced to choose the best SP with a short wait time for service. For notational convenience, we call the SP being evaluated SP_j and the SR who is using the service-community system as SR_i

The smart service community comprises N_{SP} SPs and N_{SR} SRs registered with the CU. To focus on the effect of "wait time" performance metric, we assume all N_{SP} SPs provide similar Italian food quality service, so the concern is the wait time. Each SR is modeled by a Poisson process with the average arrival rate (to select a dine-in restaurant SP) of λ_{SR}. So the collective arrival rate of SRs is $N_{SR}\lambda_{SR}$. Each SP service process is also a Poisson process with the average service rate of μ_{SP}. Further, each SP can service m_{SP} SRs simultaneously. So the collective service rate of all SPs is $N_{SP}\mu_{SP}m_{SP}$.

The wait time advertised by an SP denoted, $M_{adv}^{SP,wait_t}$, is one useful piece of information to be updated by the SP on a regular basis (minute-to-minute). It tells an SR how long the SR has to wait before being served if the SR were to arrive at the SP's location at this moment. The service time is not counted toward the wait time as the SR is not considered waiting as soon as the service is available to the SR.

An SR arriving at an SP will have to wait when the SP service capacity is full. When an SR arrives (looking for Italian food SPs online through the SCC's cloud utility), it has the choice of which SP it wants to go based on the SP's overall "subjective" trust score and SP's projected wait time received from the CU. Both the percentages of malicious SPs and SRs are P_M with the default value being 30%. We vary

P_M in the range of [10%-50%] in increment of 20% to test the sensitivity of the result w.r.t. P_M. A malicious SP advertises a false "advertised wait time" based on a risk factor percentage parameter R_f by reducing the actual wait time by R_f once it reaches the full capacity. We vary R_f in the range of [50%-100%] in increment of 25% to test the sensitivity of the result w.r.t. R_f

The actual wait time is known to an SP (good or bad) based on the arrival times and the departure times of existing SRs currently being served and the number of SRs currently waiting in the queue. A malicious SR will perform ballot-stuffing attacks by reporting a malicious SP's wait time being R_f lower than the actual wait time, which supports the advertised wait time broadcast by the malicious SP. A malicious SR also will perform bad-mouthing attacks by reporting a good SP's wait time being R_f higher than the actual wait time, which refutes the advertised wait time broadcast by the good SP. The system accepts n_r witnesses with the highest credibility scores (w.r.t. a querying SR) among all (including self) that have submitted their service rating reports to the CU for calculating the SR-SP trust score and the SP's projected service wait time.

We assume that a customer (or an SR) will select an SP for service only if the advertised wait time is less than $t_{th,advertied}$=30 minutes. After a customer selects an SP and waits for service, the customer uses the computational procedure described in Section IV.B.1 to compute the service rating for the "wait time" performance metric toward the selected SP. While in reality, a customer may not wait after the actual wait time exceeds the advertised wait time, for experimental evaluation purposes, we will allow the customer to wait until service is rendered. Using IoT-assist smart devices, an SR can automatically measure the total service time of an SR at an SP location by sensing a sudden change of the ambient environment in lighting, noise, and background music, and measure the wait time by sensing sudden changes in flavor and smell only while all others environment conditions remain the same.

The following two performance metrics (as a function of time) are considered in our comparative performance analysis:

1. An SP's projected wait time as predicted by the CU based on (17) vs. the actual wait time experienced by a good SR. The difference of these two wait time values indicates the prediction power of the system. The smaller the difference, the better the prediction power.
2. The percentage of malicious SPs selected by a good SR. This performance metric indicates the degree to which the system is trustworthy, such that malicious SPs are not likely to be selected by good SRs for service. The smaller the number, the better the trustworthiness of the system.

The performance of our trust protocol is compared against two baseline protocols:

1. Beta Reputation System [5]: Service ratings reported by individual witnesses toward an SP are combined. This trust protocol is different from our trust protocol in that an SP's trust score computation is based on common belief or reputation in nature, while our trust protocol is one-to-one subjective trust evaluation in nature. However, both approaches are trust-based. Therefore, there is no differences in the projected wait time calculation (see Section IV.B.4) or application-level user decision making (see Section IV.B.5 and Section IV.C).
2. Non-Trust-based: An SR simply selects an SP for which it had the best prior service rating. If there is no prior service experience or it only had negative service experience, an SR randomly selects one SP out of the remaining eligible SPs for service.

A. PROTOCOL PERFORMANCE

We first evaluate accuracy, convergence and resiliency properties of our trust protocol. Later in Section V.B. we perform a comparative analysis of our trust protocol against the baseline protocols.

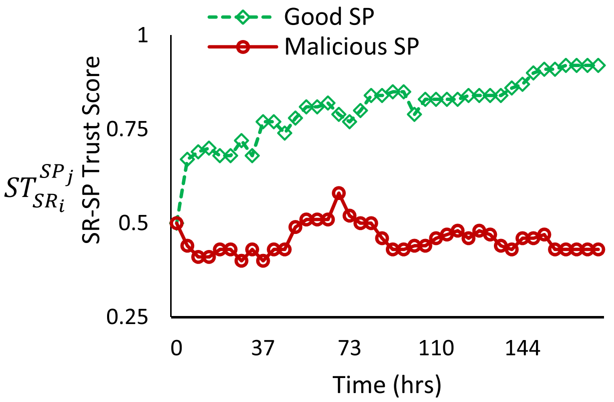


FIGURE 6. Subjective trust scores of a good SP (green curve) and a malicious SP (red curve) vs. time.

Fig. 6 shows the subjective trust scores of a malicious SP with risk factor $R_f = 100\%$ and of a good SP (both arbitrarily chosen) from the perspective of a good SR (also arbitrary chosen) as time progresses to demonstrate accuracy, convergence and resiliency properties of our trust-based service community management system.

In Fig. 6, the X coordinate marks the time points at which service transactions occurred in the system and the Y coordinate marks $SR_i{}'s$ subjective trust score toward $SP_j(ST_{SR_i}^{SP_j})$as computed by (16). When the system starts, the trust score for both malicious and good SPs goes up because initially all SPs are free and the wait time is always less than 20% of the projected wait time (corresponding to a rating of 5). However, as time progresses, more SRs gather positive/negative service experiences based on the actual wait time observed. In case of the malicious SP, since the actual wait time will be much higher than the advertised wait time, many SRs log negative experience about the malicious SP. As a result, the trust score goes down. For the good SP, the trust score goes up because many SRs log positive experiences.

As time progresses, the trust score of the good SP converges to 0.9 because the customers observed that the actual wait time is within 20% of the good node's advertised wait time (corresponding to a rating of 4 to 5). On the other hand, the trust score of the malicious SP converges to close to 0.4 because the customers observed that the actual wait time is 60%-100% (corresponding to a rating of 1 to 2) over the malicious SP's advertised wait time whose $R_f = 100\%$. We attribute the success in filtering malicious SPs to our protocol's ability to effectively filter out malicious recommenders. As a querying SR's self-experience improves as time progresses, it is able to effectively filter out malicious recommenders (30% witnesses are malicious in this scenario), due to a more accurate assessment of a malicious SR's credibility. We conclude that our trust protocol is effective in trust accuracy convergence, and resilience against malicious attacks.

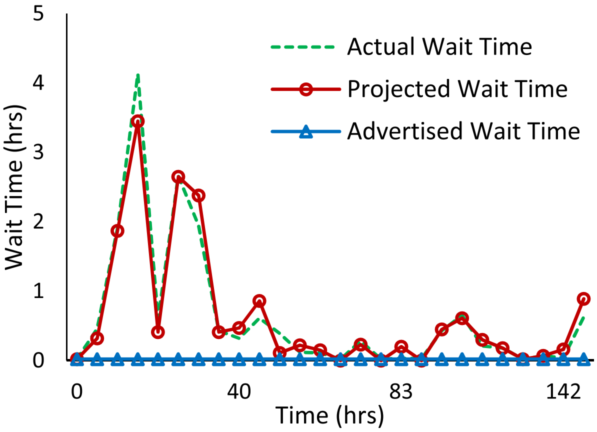


FIGURE 7. A malicious SP's projected wait time (blue curve), advertised wait Time (green curve), and actual wait time (red curve).

Fig. 7 shows the projected wait time $M_{SR_i}^{SP_j,wait_t}$ as predicted from (17) vs. the advertised wait time $M_{adv}^{SP_j,wait_t}$ and the actual wait time of a malicious SP (arbitrarily chosen) experienced by all SRs over time, with each data point representing a wait time value experienced by an SR.

We first observe that the plot is a wait time vs. time plot. The wait time is up and down because the arrival process of customers is stochastic, so customers may arrive at around the same time in which case the wait time would be high, or customers may arrive in an interleaving fashion in which case the wait time would be low. In either case, we observe that the difference of the projected wait time and the actual wait time is very small. This demonstrates the accuracy of our protocol. Also the projected wait time is much higher than this malicious SP's own advertised wait time. The reason is that this malicious SP advertises $M_{adv}^{SP_j,wait_t} = 0$ (as it reduces the wait time by $R_f = 100\%$) to attract customers, but the wait time is actually much higher. We observe that, for this malicious SP, the projected wait time curve in Fig. 7 matches the SP trust score curve in Fig. 6 because the trustworthiness of this malicious SP goes down as more witnesses (including the querying SR itself) report bad wait time experiences to the CU who uses cumulative trust evidence for the projected wait time calculation. The cumulative trust evidence itself is based on selecting n_r witnesses ($n_r = 5$ for this scenario) with the highest credibility among all witnesses thus ensuring the wait time values reported by SR's with the high credibility will dominate the resulting projected wait time (as computed by (17)).

Fig. 8 shows the percentage of malicious SPs selected to provide service for a good SR (randomly selected) as time progresses. There are three curves corresponding to the three user decision policies discussed in Section IV.B.5, namely, "select the least wait time SP," "select the most trustworthy SP," and "select the least wait time SP among trustworthy SPs," respectively.

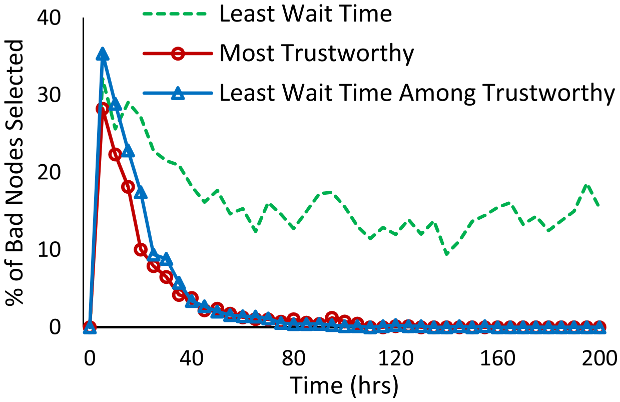


FIGURE 8. Percentage of malicious SPs selected for service over time by a good SR under Least Wait Time (green curve), Most Trustworthy (red curve) and Least Wait Time Among Trustworthy (blue curve).

Fig. 8 shows that under the "select least wait time" policy, the percentage of malicious nodes selected is highest in comparison with "select the most trustworthy SP" and "select the least wait time SP among trustworthy SPs." The reason is that in the "select least wait time SP" policy, SP's are selected solely based on wait time without considering the maliciousness of the selected SP. An SP is selected as long as the projected wait time is the lowest among all. Consequently, it has the highest chance of selecting malicious nodes. On the other hand, the "select the most trustworthy SP" policy selects the SP with the highest trust score among all (i.e., with the highest probability of not being malicious). Consequently, it has the least chance of selecting malicious SPs for service. For this scenario in which the percentage of malicious nodes is 30%, the "select the least wait time SP among trustworthy SPs" policy performs just as good as the "select the most trustworthy SP" policy because there are enough trustworthy SPs (70%) to select from.

Fig. 9 shows the actual wait time experienced by a good SR (randomly selected) as time progresses. There are three curves corresponding to the three user decision policies. We observe that the wait time is the highest under the "select the most trustworthy SP" policy. The reason is that the

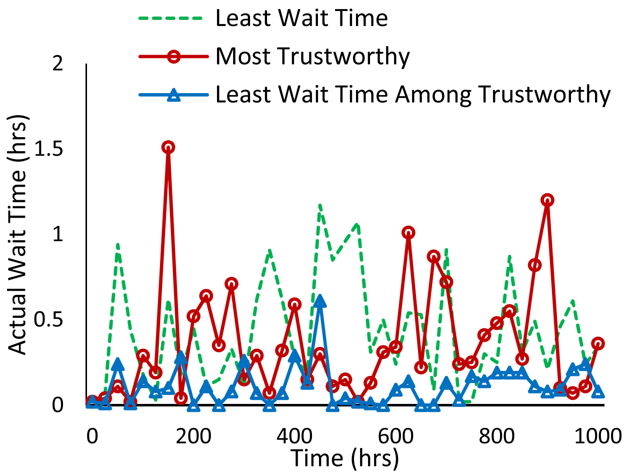


FIGURE 9. Comparison of actual wait time experienced by a good SR under Least Wait Time (green curve), Most Trustworthy (red curve) and Least Wait Time Among Trustworthy (blue curve).

"select the most trustworthy SP" policy (red curve) does not care if the SP selected will provide the least wait time and it only selects the SP with the highest trust score among all (i.e., with the highest probability of not being malicious) without considering the wait time, thus it tends to incur the highest wait time compared to the other two policies. We observe that the "select the least wait time SP among trustworthy SPs" policy (blue curve) performs just as good as (and frequently is even better than) the "select least wait time" policy (green curve) in terms of the actual wait time because the projected wait time predicted by the CU is accurate, even if the SP selected is malicious, w.r.t. the actual wait time (as demonstrated in Fig. 7 earlier).

From comparing Fig. 8 with Fig. 9, we see that the "select the least wait time SP among trustworthy SPs" policy can best balance "the service wait time" performance metric with "the percentage of malicious nodes selected" performance metric by adjusting the magnitude of the minimum trust threshold ST_{th}, i.e., when $ST_{th} = 0$ it degenerates to "select the least wait time SP" and when $ST_{th} = 1$ it degenerates to "select the most trustworthy SP."

B. COMPARITIVE ANALYSIS

In this section, we perform a comparative analysis of our protocol against Josang's Beta Reputation System trust protocol [5] and the non-trust-based protocol. Fig. 10 (corresponding to Fig. 8) shows the percentage of malicious SPs selected by a good SR as time progresses with $P_M = 30\%$ and $R_f = 100\%$ for the proposed service community. The blue curve labeled with "TMSSC" is for the SSC with our trust management protocol in place under the "select the least wait time SP among trustworthy SPs" policy. The red curve labeled with "Beta Reputation System" is for the service community with Josang's Beta Reputation System trust protocol [5] in place. The green curve labeled "Non-Trust-based" is for the service community without a trust protocol in place.

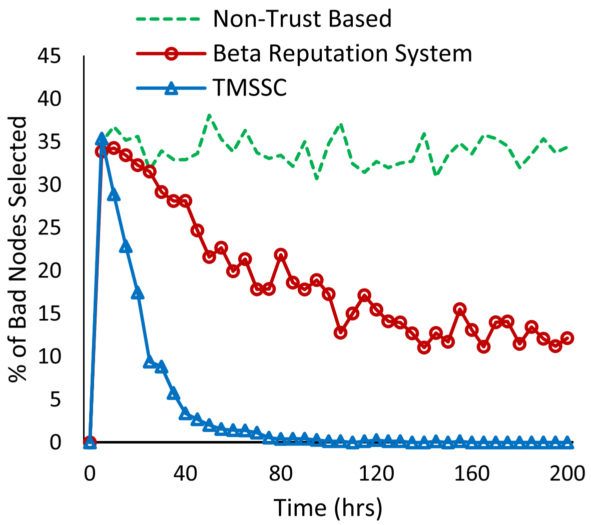


FIGURE 10. Comparison of the percentage of malicious SP selected by a good SR under Non-Trust Based (green curve), Beta Reputation System (red curve), and our proposed TMSSC protocol (blue curve).

With our TMSSC protocol in place (blue curve), the service community is able to identify malicious SPs over time. Initially all SPs have a trust score of 0.5 (to deal with the cold start problem), so SRs initially could still select malicious SPs for service. However, as time progresses, the trust score of malicious SPs goes down due to negative service experiences gathered by witnesses. Consequently, the projected wait time for malicious SPs would become much higher than the advertised wait time. As a result, malicious SPs would not be selected for service because they are identified as untrustworthy (i.e., $ST_{SR_i}^{SP_j} < ST_{th}$)and the percentage of malicious SPs selected to provide service goes down quickly as time progresses.

With Josang's Beta Reputation System trust protocol in place (red curve), the system is also able to identify malicious SPs over time, albeit at a slower rate because malicious witnesses' ratings are combined without filtering. Since the percentage of good witnesses is much higher than the percentage of malicious nodes, a reputation system based on public opinions or common belief can still maintain a healthy service community. However, we can see our TMSSC protocol outperforms Josang's Beta Reputation System trust protocol by a wide margin because we take one-to-one, subjective witness credibility into consideration to effectively filter out low-credibility service rating reports to form the overall trust score (i.e., via (18)) instead of relying on common belief reputation systems which are susceptible to collusion and are dependent on the availability of a high percentage of good witnesses.

The last case is for "Non-Trust-based" (the green curve). For a service community without a trust protocol in place, an SR needs to depend on its prior service experience, and thus initially can only take the advertised wait time as the projected wait time for decision making. So initially the percentage of malicious SPs selected is high. However, the percentage of malicious SPs selected to provide service decreases over time because an SR uses its own experience to select an SP for which the wait time service rating is high (i.e. selects a visited SP for which it had the best prior service rating).

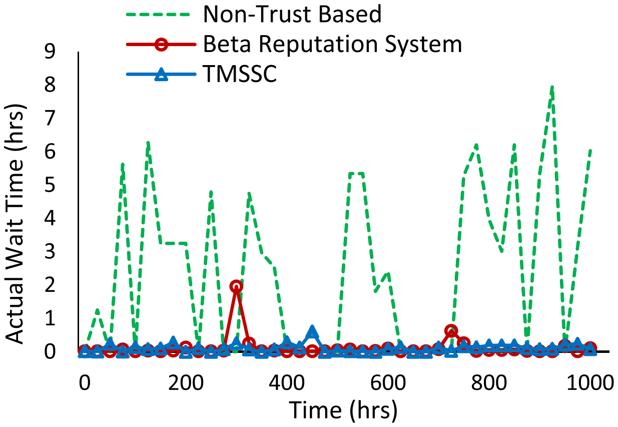


FIGURE 11. Comparison of actual wait time experienced by a good SR under Non-Trust Based (green curve), Beta Reputation System (red curve), and our TMSSC protocol (blue).

Fig. 11 (corresponding to Fig. 9) shows the actual wait time experienced by a good SR as time progresses with $P_M = 30\%$ and $R_f = 100\%$ in the smart Italian food service community under three protocols. The main reason for the performance difference depends on whether this SR is able to select SPs for service such that not only the "SP's projected wait time" is small (thus incurring a small actual wait time), but also the "SP's advertised wait time" is about the same as the "SP's projected wait time" (thus selecting only trustworthy SPs). Unlike the two baseline protocols, our proposed TMSSC protocol is capable of maintaining accurate one-to-one SR-SP and SR-SR trust scores and thus provide accurate projected wait times (by means of (17)) for SP-selection decision making. As a result, we observe that our trust protocol outperforms both Josang's Beta Reputation System trust protocol and the non-trust-based baseline protocol in terms of the wait time performance metric, which is of ultimate importance to users in this smart food service community.

VI. CONCLUSION

In this paper, we proposed the notion of a "smart service community" (SSC) consisting of service providers (SPs), service requestors (SRs), and a cloud utility for a specific type of service typically in a smart city setting. We proposed novel trust-based service management designs to support this new notion. We investigated ways to utilize IoT-assisted technology for individual SRs to automatically measure one-to-one SR-SP service ratings and SR-SR credibility ratings and for the cloud utility to integrate them together into one-to-one subjective SR-SP trust scores for each pair of SR and SP in the system, such that an SR can subjectively select what it believes to be the best or most trustworthy SP among all available for service based on its own performance metrics. Our trust-based service management designs effectively fend off recommendation attacks and collusive attacks even if the majority of recommenders are malicious. We demonstrated the effectiveness of our trust-based SSC design over contemporary service ranking systems via a smart Italian food service community.

As future work, we plan to extend our work in terms of (1) dealing with more sophisticated attacks (e.g.,..[27], [29]–[31]); and (2) developing decentralized or distributed systems allowing the load of trust aggregation to be distributed over individual components of the system for better applicability.

ACKNOWLEDGMENT

The authors would like to thank Kshitij Karki for helping with coding and debugging SMPL programs for experimental evaluation.

REFERENCES

[1] H. Al-Hamadi and I. R. Chen, "Trust-based decision making for health IoT systems," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1408–1419, Oct. 2017.

[2] M. Eirinaki, M. D. Louta, and I. Varlamis, "A trust-aware system for personalized user recommendations in social networks," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 44, no. 4, pp. 409–421, Apr. 2014.

[3] J. Timpner, D. Schürmann, and L. Wolf, "Trustworthy parking communities: Helping your neighbor to find a space," *IEEE Trans. Dependable Secure Comput.*, vol. 13, no. 1, pp. 120–132, Jan./Feb. 2016.

[4] Google Cloud. (2018). *Geo-Location APIs | Google Maps Platform | Google Cloud*. Accessed: Nov. 4, 2018. [Online]. Available: https://cloud.google.com/maps-platform/

[5] A. Josang and R. Ismail, "The beta reputation system," in *Proc. 15th Bled Electron. Commerce Conf.*, vol. 5, 2002, pp. 2502–2511.

[6] G. Wang, B. Wang, T. Wang, A. Nika, H. Zheng, and B. Y. Zhao, "Ghost riders: Sybil attacks on crowdsourced mobile mapping services," *IEEE/ACM Trans. Netw.*, vol. 26, no. 3, pp. 1123–1136, Jun. 2018.

[7] K. Zhang, X. Liang, R. Lu, and X. Shen, "Sybil attacks and their defenses in the Internet of Things," *IEEE Internet Things J.*, vol. 1, no. 5, pp. 372–383, Oct. 2014.

[8] I.-R. Chen, F. Bao, and J. Guo, "Trust-based service management for social Internet of Things systems," *IEEE Trans. Dependable Secure Comput.*, vol. 13, no. 6, pp. 684–696, Nov./Dec. 2016.

[9] I.-R. Chen, J. Guo, and F. Bao, "Trust management for SOA-based IoT and its application to service composition," *IEEE Trans. Services Comput.*, vol. 9, no. 3, pp. 482–495, May/Jun. 2016.

[10] M. Nitti, L. Atzori, and I. P. Cvijikj, "Friendship selection in the social Internet of Things: Challenges and possible strategies," *IEEE Internet Things J.*, vol. 2, no. 3, pp. 240–247, Jun. 2015.

[11] J. Guo, I.-R. Chen, and J. J. P. Tsai, "A survey of trust computation models for service management in Internet of Things systems," *Comput. Commun.*, vol. 97, pp. 1–14, Jan. 2017.

[12] M. Nitti, R. Girau, and L. Atzori, "Trustworthiness management in the social Internet of Things," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 5, pp. 1253–1266, May 2014.

[13] W. Feng, Z. Yan, H. Zhang, K. Zeng, Y. Xiao, and Y. T. Hou, "A survey on security, privacy, and trust in mobile crowdsourcing," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 2971–2992, Aug. 2018.

[14] C. Prandi, S. Mirri, S. Ferretti, and P. Salomoni, "On the need of trustworthy sensing and crowdsourcing for urban accessibility in smart city," *ACM Trans. Internet Technol.*, vol. 18, no. 1, 2017, Art. no. 4.

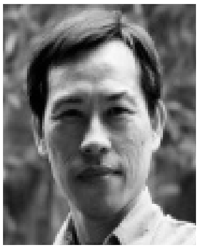
[15] C. Prandi, P. Salomoni, and S. Mirri, "mPASS: Integrating people sensing and crowdsourcing to map urban accessibility," in *Proc. IEEE Int. Conf. Consum. Commun. Netw. Conf.*, Jan. 2014, pp. 10–13.

[16] K. Mo, E. Zhong, and Q. Yang, "Cross-task crowdsourcing," in *Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2013, pp. 677–685.

[17] G. McArdle and R. Kitchin, "Improving the veracity of open and real-time urban data," *Built Environ.*, vol. 42, no. 3, pp. 457–473, 2016.

[18] M. N. Vidya and P. K. S. Nandini, "Adaptation trust based protocol for IoT using smartphones in social media: Travel map guide," in *Proc. IEEE 2nd Int. Conf. Appl. Theor. Comput. Commun. Technol. (iCATccT)*, Jul. 2016, pp. 109–113.
[19] Z. Lin and L. Dong, "Clarifying trust in social Internet of Things," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 2, pp. 234–248, Feb. 2018.
[20] M. Nitti, R. Girau, L. Atzori, and V. Pilloni, "Trustworthiness management in the IoT: The importance of the feedback," in *Proc. IEEE 20th Conf. Innov. Clouds, Internet Netw. (ICIN)*, Mar. 2017, pp. 325–327.
[21] H. Xia, Z. Li, Y. Zheng, A. Liu, Y. Choi, and H. Sekiya, "A novel light-weight subjective trust inference framework in MANETs," *IEEE Trans. Sustain. Comput.* doi: 10.1109/TSUSC.2018.2817219.
[22] X. Wu, B. Cheng, and J. Chen, "Collaborative filtering service recommendation based on a novel similarity computation method," *IEEE Trans. Services Comput.*, vol. 10, no. 3, pp. 352–365, May 2017.
[23] K. Su, B. Xiao, B. Liu, H. Zhang, and Z. Zhang, "TAP: A personalized trust-aware QoS prediction approach for Web service recommendation," *Knowl.-Based Syst.*, vol. 115, pp. 55–65, Jan. 2017.
[24] O. Ben Abderrahim, M. H. Elhedhili, and L. Saidane, "CTMS-SIOT: A context-based trust management system for the social Internet of Things," in *Proc. IEEE 13th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2017, pp. 1903–1908.
[25] S. Ding, Y. Li, D. Wu, Y. Zhang, and S. Yang, "Time-aware cloud service recommendation using similarity-enhanced collaborative filtering and ARIMA model," *Decis. Support Syst.*, vol. 107, pp. 103–115, Mar. 2018.
[26] P. Resnick, R. Zeckhauser, E. Friedman, and K. Kuwabara, "Reputation systems," *Commun. ACM*, vol. 43, no. 12, pp. 45–48, 2000.
[27] R. Mitchell and I. R. Chen, "Modeling and analysis of attacks and counter defense mechanisms for cyber physical systems," *IEEE Trans. Rel.*, vol. 65, no. 1, pp. 350–358, Mar. 2016.
[28] M. H. MacDougall, *Simulating Computer Systems*. Cambridge, MA, USA: MIT Press, 1987.
[29] H. Al-Hamadi and I.-R. Chen, "Adaptive network management for countering selective capture in wireless sensor networks," in *Proc. 9th Int. Conf. Netw. Service Manage.*, Zürich, Switzerland, 2013, pp. 203–210.
[30] J. Caminha, A. Perkusich, and M. Perkusich, "A smart middleware to detect on-off trust attacks in the Internet of Things," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2018, pp. 1–2.
[31] C. V. L. Mendoza and J. H. Kleinschmidt, "Defense for selective attacks in the IoT with a distributed trust management scheme," in *Proc. IEEE Int. Symp. Consum. Electron. (ISCE)*, Sep. 2016, pp. 53–54.

HAMID AL-HAMADI received the B.S. degree in information technology from Griffith University, Brisbane, Australia, in 2003, the M.S. degree in information technology from the Queensland University of Technology, Brisbane, Australia, in 2005, and the Ph.D. degree in computer science from the Virginia Polytechnic Institute and State University, Blacksburg, VA, USA, in 2014. He has worked for Universe Computers as an on-site Network Engineer with Kuwait National Petroleum Company. He has worked as a Core Network Engineer with Tawasul Telecom, Kuwait. He is currently an Assistant Professor with the Department of Computer Science, Kuwait University, Kuwait. His current research interests include the Internet of Things, security, mobile cloud, trust management, and reliability and performance analysis.



ING-RAY CHEN received the B.S. degree from National Taiwan University, Taiwan, and the M.S. and Ph.D. degrees in computer science from the University of Houston, Houston, TX, USA. He is currently a Professor with the Department of Computer Science, Virginia Tech. His research interests include mobile computing, wireless systems, security, trust management, and reliability and performance analysis. He is a recipient of the IEEE Communications Society William R. Bennett Prize in the field of Communications Networking. He currently serves as an Editor for the IEEE Transactions on Services Computing, the IEEE Transactions on Network and Service Management, and *The Computer Journal*.

JIN-HEE CHO (S'07–M'09–SM'14) received the M.S. and Ph.D. degrees in computer science from Virginia Tech, in 2004 and 2008, respectively. She is currently an Associate Professor with the Department of Computer Science, Virginia Tech. Prior to joining the Virginia Tech, she has been a Computer Scientist with the U.S. Army Research Laboratory, Adelphi, MA, USA, since 2009. She has published over 100 peer-reviewed technical papers in leading journals and conferences in the areas of trust management, cybersecurity, metrics and measurements, network performance analysis, resource allocation, agent-based modeling, uncertainty reasoning and analysis, information fusion/credibility, and social network analysis. She is a member of the ACM. She has received the Best Paper Aw ards in the IEEE TrustCom'2009, the BRIMS'2013, the IEEE GLOBECOM'2017, the 2017 ARL's Publication Award, and the IEEE CogSima 2018. She is a winner of the 2015 IEEE Communications Society William R. Bennett Prize in the Field of Communications Networking. In 2016, she was selected for the 2013 Presidential Early Career Award for Scientists and Engineers.