TrustMeter: A Trust Assessment Scheme for Collaborative Privacy Mechanisms in Participatory Sensing Applications

Delphine Christin*†, Daniel Rodriguez Pons-Sorolla‡, Matthias Hollick‡, Salil S. Kanhere§
* Institute of Computer Science IV, University of Bonn, Bonn, Germany, † Fraunhofer FKIE, Wachtberg, Germany
‡Secure Mobile Networking Lab, Technische Universität Darmstadt, Darmstadt, Germany
§School of Computer Science and Engineering, University of New South Wales, Sydney, Australia
Emails: *christin@cs.uni-bonn.de, mhollick@seemoo.tu-darmstadt.de, salilk@cse.unsw.edu.au*

***Abstract*—In typical participatory sensing applications, mobile devices record a variety of sensor readings (e.g., sound samples and accelerometer data), which are tagged with spatiotemporal information and uploaded to an application server. The collection of detailed location data reveal insights about the users' whereabouts and daily routines, therefore seriously compromising their privacy. Users can mutually preserve their privacy by opportunistically exchanging sensor readings during physical meetings, thus breaking the link between the collected data and their permanent identities. The success of this procedure depends on the collaboration of all participating users. Our paper proposes a scheme called TrustMeter to assess the individual user contribution to this privacy protection mechanism. Based on peer-based ratings, our system attributes trust levels to each user allowing to readily identify and quarantine malicious users. We investigate the TrustMeters performance under different attacks by means of extensive simulations, and show that it succeeds in quarantining malicious users in most analyzed scenarios.**

I. Introduction

In recent years, mobile phones have been leveraged as sensing platforms due to the plethora of on-board sensors, wireless technologies, and complex processing capabilities. These features have contributed to the emergence of a myriad of participatory sensing applications, in which volunteers collect sensor readings from the surrounding environment in unprecedented quantity using their mobile phones. In virtually all participatory sensing applications, the sensor readings are stamped with time and location coordinates that define the context of their collection [1]. Without any protection mechanisms, these contextual data can leak privacy-sensitive user information out, revealing routinely visited locations, place of residence and work, etc. Mechanisms preserving user location privacy are hence mandatory to mitigate these privacy threats.

Most of the existing privacy-preserving schemes in participatory sensing rely either on the application used or a trusted third party [1]. In contrast, our collaborative path-hiding mechanism presented in [2] proposes that users mutually preserve their privacy in a decentralized fashion. Inspired from mix networks [3], users exchange previously collected sensor readings when they physically encounter each other. The jumbled sensor readings are then uploaded to the application server. Swapping a subset of the data samples removes the association between the sensor readings and the identity of the users who collected them. The sensor readings no longer reveal the actual paths followed by each user, but jumbled paths instead. However, malicious users may attempt to tamper with the normal operation of the swap mechanism, e.g., dropping exchanged data. This might lead to a total failure of the reporting mechanism and thus of the application. To overcome these threats, we make the following contributions:

1) We propose TrustMeter that assesses the behavior of users in collaborative privacy mechanisms. Malicious users are identified based on their low trust levels and subsequently excluded from the data jumbling process.
2) We implement the TrustMeter scheme in the ONE simulator [4] and investigate its performance by means of thorough simulations. The analysis of various attack scenarios shows that TrustMeter identifies most malicious users after only few exchanges.

Our paper is structured as follows. In Sec. II, we present our assumptions and models, before describing TrustMeter in Sec. III. We assess its performance in identifying attackers in Sec. IV and discuss its robustness against threats to reputation in Sec. V. Finally, we survey related work in Sec. VI, before concluding this paper in Sec. VII.

II. System and Threat Models

We first describe our system and threat models. In particular, we provide an overview of the concept of collaborative path hiding, which we introduced in [2] and serves as the sample application to demonstrate the efficacy of our scheme.

A. System Model

We assume participatory sensing applications without real-time constraints for data delivery. Mobile devices automatically collect sensor readings. Each sensor reading s is stamped with the collection time t and location information l forming a triplet (t, l, s). Instead of directly reporting the triplets to the application server, the mobile devices leverage the collaborative path-hiding concept. This concept introduced in [2] aims at breaking the association between the spatiotemporal context (i.e., time and location) at which the sensor readings were

taken, and the user identity. This is achieved by physically exchanging triplets between users in an opportunistic fashion. The mobile devices can exchange triplets according to one of the following exchange strategies: (1) *realistic strategy:* users exchange the entire set of collected/exchanged triplets with a certain probability, (2) *random-unfair strategy:* each user independently and randomly determines the number of triplets to exchange, (3) *random-fair strategy:* users agree on a common number of triplets to exchange at each meeting. Each user periodically uploads a combination of the exchanged triplets and their own collected triplets to the server. The server runs a function, such as applied in [7], that identifies and discards erroneous triplets. Based on this function, the server attributes a reputation value to each user, which is later used to weight her results in the computation of the application summaries.

B. Threat Model

As an artifact of the collaborative nature of the path-hiding mechanism, users can become adversaries. Malicious users can attempt to breach other users' privacy by directly accessing the information included in the exchanged triplets. They can also either drop exchanged triplets or create false triplets to exchange in order to alter the results consolidated at the server side, and perturb the function of the collaborative path-hiding mechanism. In what follows, we especially focus on these adversaries. We refer to malicious users who neither exchange nor report triplets from other users as *droppers*, and to users who create falsified triplets as *spammers*.

We further assume curious-but-honest application administrators who attempt to passively breach the participant privacy, but run the system normally and faithfully. They, however, do not launch active attacks (such as collusions with users or reputation manipulation) to obtain further information. Note that we have analyzed the administrators' capabilities in identifying exchanged from original triplets in [5].

Our adversaries follow the Dolev-Yao threat model [6], i.e., they are able to listen to all communication, fabricate, replay, and destroy messages. However, they are not able to compromise cryptographic mechanisms.

III. TrustMeter Architecture

TrustMeter relies on the building blocks depicted in Fig. 1 to identify and quarantine malicious clients (see Sec. II-B). In what follows, we assume that all clients have a unique client ID and each party has a public/private key pair.

A. Overview

Each client periodically transmits the triplets to the server. At each transmission, the server disseminates a table containing the trust level of all users. The trust level is computed by the server based on the users' reports. When a client encounters another client, the two entities first examine the trust level of the opposite party provided by the server. Only if both clients are not rated as *untrusted*, they initiate an exchange using one of the three strategies (cf. Sec. II-A).

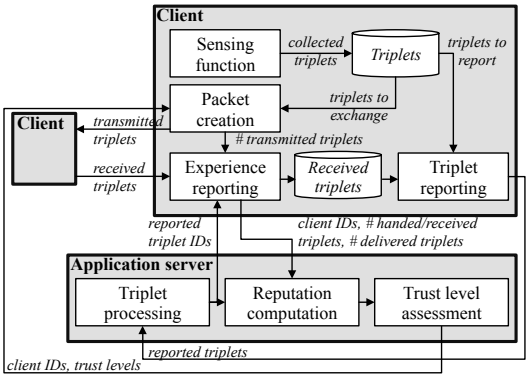


Fig. 1. Overview of the TrustMeter architecture

B. Underlying Mechanisms

1) Preparation and Exchange of Triplets: After having verified the trustworthiness of the opposite party, each client individually encrypts the triplets using the server's public key in order to prevent other clients from accessing its content. For each triplet, the client generates a unique triplet ID by concatenating (1) its client ID, (2) the triplet timestamp, and (3) a random number. It then computes a one-way hash function of the result. Other mechanisms to generate unique triplet IDs can however be used without loss of generality. Before exchanging these triplets, the client stores their IDs and the ID of its exchange partner. It also sets a delivery timeout specific to the exchanged triplets. If these triplets do not reach the server before the timeout expiration, the clients consider them as lost and exchange them again.

2) Reporting of Triplets: When reporting triplets to the server, clients track if their triplets have been delivered to the server and report their experience with other clients as follows.

a) Tracking of Triplets: The clients request the list of the triplet IDs delivered to the server since their last connection. Based on this list, the clients determine which triplets have been successfully delivered and remove them from their internal storage. For the remaining triplets, the clients wait until the expiration of the timeout to retransmit them. Note that clients cannot directly request the IDs of interest, since the server would establish a link between the reported triplets and the identity of the client having collected them.

b) Experience Reporting: The clients report their experience with all exchange partners by compiling statistics for each pair-wise exchange. These statistics include the IDs of the two entities involved, the number of triplet exchanges in both directions, and the number of triplets delivered to the server. Note that neither the IDs of the exchanged triplets nor the fraction of own collected triplets are disclosed to the server.

3) Reputation Computation and Trust Assessment: Based on the reported triplets and experiences, the server computes the reputation score and derives the associated trust level for each client. To this end, we introduce two reputation scores, R and R'. R measures the clients' participation based on the numbers of exchanged and delivered triplets, and hence provides insights about potential dropping behavior. In

contrast, R' identifies potential spamming behavior based on the reported triplets. For each new client, the server initializes R with R_0 and R' with R'_0 in order to allow new clients to participate in exchanges with already active clients.

R is recursively computed according to Eq. 1, which corresponds to an exponential moving average. $R \in$ [0,100] and δ is the ratio of the number of delivered triplets to the total number of exchanged triplets. The lower the value of δ, the higher the probability that the client has dropped packets.

$$R_{n+1} = R_n + \frac{100 \cdot \delta - R_n}{4} \tag{1}$$

For R', the server can detect falsified triplets referred to as spam (see Sec. II-B). When it happens, the server flags the client having reported these triplets as well as its exchange partners. The server computes two metrics as indicator of spamming behavior: γ_1 is the ratio of the number of flagged exchanges over the total number of exchanges and γ_2 is the ratio of the number of reported triplets identified as spam over the total number of reported triplets. The higher γ_1 and γ_2, the higher the probability that the clients have injected spam. Based on these metrics, the server computes $R^{(1)}, R^{(2)}, R^{(3)} \in$ [0,100] as follows, γ being the mean of γ_1 and γ_2.

$$R^{(1)}_{n+1} = R^{(1)}_n + \frac{100 \cdot (1-\gamma_1) - R^{(1)}_n}{4} \tag{2}$$

$$R^{(2)}_{n+1} = R^{(2)}_n + \frac{100 \cdot (1-\gamma_2) - R^{(2)}_n}{4} \tag{3}$$

$$R^{(3)}_{n+1} = R^{(3)}_n + \frac{100 \cdot (1-\gamma) - R^{(3)}_n}{4} \tag{4}$$

Next, the server computes R' according to Alg. 1. λ_U and λ_T are the reputation thresholds used by the server to classify the clients into the categories introduced in Tab. I. The rationale behind this algorithm is to first identify clients that are clearly malicious (i.e., $R^{(3)} < \lambda_U$) or honest (i.e., $R^{(3)} > \lambda_T$). Then, the classification is refined for clients that are still considered as *indefinite*. Steps 2 to 3 in Alg. 1 target the last honest clients based on the value of $R^{(2)}$, while steps 4 to 5 focus on the last malicious clients based on the value of $R^{(1)}$. Finally, the server selects the minimum of R_{n+1} and R'_{n+1} as reputation value and associates the corresponding trust level to the client according to Tab. I. Choosing the minimum prevents attackers from masking spamming behavior by well behaving in terms of dropping and vice versa.

4) Dissemination of Trust Levels: When a client reports triplets to the server, it obtains an update of the trust levels of all other clients. Clients leverage these levels to decide to exchange triplets with new encountered clients as introduced in Section III-B1.

IV. Performance Analysis

We next analyze the performance of TrustMeter in identifying malicious behaviors (i.e., spamming and dropping) by means of extensive simulations.

Algorithm 1 Computation of R'

1: **if** $\lambda_U \leq R^{(3)} \leq \lambda_T$ **and** $R^{(2)} > \lambda_T$) **then**
2: $\quad R' \leftarrow R^{(2)}$
3: **else if** ($\lambda_U \leq R^{(3)} \leq \lambda_T$ **and** $R^{(1)} < \lambda_U$) **then**
4: $\quad R' \leftarrow R^{(1)}$
5: **else**
6: $\quad R' \leftarrow R^{(3)}$
7: **end if**
8: **return** R'

TABLE I
Mapping of trust levels with reputation values

Trust level	Reputation range
Untrusted	$0 \leq min(R, R') < \lambda_U$
Indefinite	$\lambda_U \leq min(R, R') \leq \lambda_T$
Trusted	$\lambda_T < min(R, R') \leq 100$

A. Simulation Setup and Method

We have implemented the TrustMeter scheme in the ONE simulator [4]. We consider a constant population of 100 clients over 24 hours for each simulation. The clients follow a pedestrian movement model and generate a new triplet every 150 s. They report the triplets to the server on a hourly basis. We assume an ideal communication environment and a communication radius of 10 m consistent with typical Bluetooth communication range. We further assume that clients exchange triplets only with clients rated as *indefinite* or *trusted* using one of the following strategies: (1) realistic, (2) random-unfair, and (3) random-fair. We set the system parameters as follows: (1) α=$\frac{1}{4}$, (2) R_0=R'_0=50, (3) λ_U=30 and λ_T=70, and (4) timeout to two hours. Note that the selection of these values is extensively discussed in [8].

As performance metric, we use the Matthews Correlation Coefficient (MCC) defined as:

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP) \cdot (TP+FN) \cdot (TN+FP) \cdot (TN+FN)}} \tag{5}$$

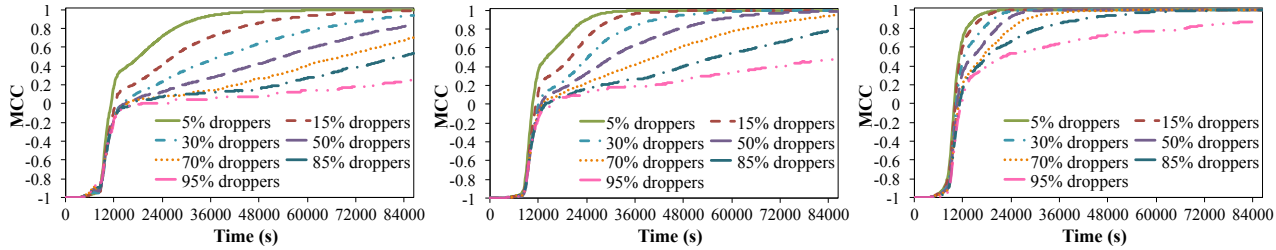
where *TP* is the number of true positives (attackers identified as attackers), *TN* is the number of true negatives (honest clients identified as such), *FP* is the number of false positives (honest clients identified as attackers) and *FN* the number of false negatives (attackers identified as honest clients). We consider attackers and honest clients that are classified as *indefinite* as false negatives and false positives, respectively.

The MCC ranges between -1 and +1. A value of +1 indicates a perfect identification of both attackers and honest clients, while a value of -1 indicates a total failure in identifying both categories. We repeat each simulation 50 times using different random seeds to model different movement patterns. We present the averaged results in the next sections.

B. Droppers

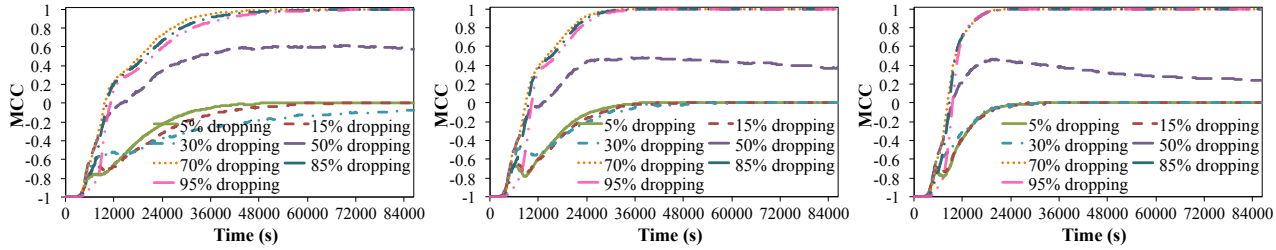
We first investigate the TrustMeter performance in identifying different percentages of droppers and dropping rates.

1) Percentage of Droppers: Fig. 2 represents the temporal evolution of the MCC for different percentages of droppers dropping all triplets. We can observe an initial bootstrapping phase followed by a steady state. The initial phase provide



(a) Realistic strategy (b) Random-unfair strategy (c) Random-fair strategy

Fig. 2. MCC for different percentages of droppers and exchange strategies over time



(a) Realistic strategy (b) Random-unfair strategy (c) Random-fair strategy

Fig. 3. MCC for different dropping rates and exchange strategies over time

insights about the sensitivity of the scheme, as each triplet significantly impacts the reputation of the clients at the beginning. After this phase, the system stabilizes and assigns a definitive trust level to the client. This process takes some time since the trust levels are derived from observations of the past client behavior.

Overall, the lower the percentage of droppers, the faster their identification, as all clients' reputation is affected by the dropping action. The fewer droppers, the greater increase in reputation, allowing honest clients to reach faster the *trusted* level. The identification of droppers is faster for the random-fair strategy than for the random-unfair and realistic strategies. Because clients exchange fewer triplets with the random-fair strategy, the impact of triplet drop on the reputation of honest clients is lower than with the other strategies.

2) Dropping Rate: Next, we assume a population of 90 honest clients and ten droppers and vary the dropping rate on the performance. Fig. 3 shows that TrustMeter performs poorly in distinguishing honest clients from droppers for dropping rates below or equal to 30%. For these dropping rates, there is a lower packet drop. This slows down the reputation reduction process, and it takes longer labeling a dropper as *untrusted*.

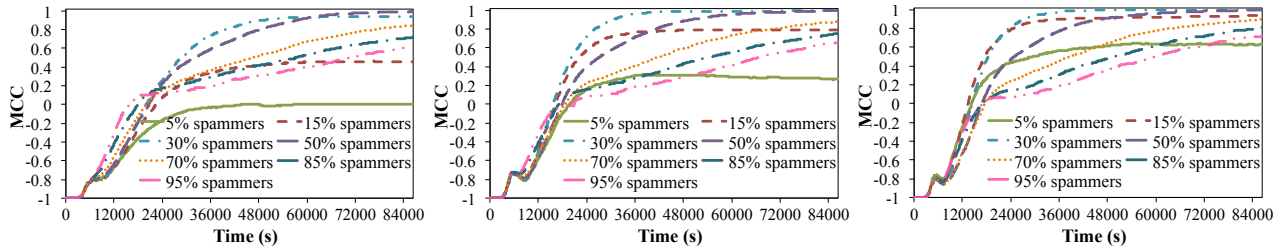
However, the performance improves for 50% and above dropping rates, with MCC values around 0.6 for the realistic strategy (see Fig. 3(a)), 0.4 for the random-unfair strategy (see Fig. 3(b)) and 0.2 for the random-fair strategy (see Fig. 3(c)). For dropping rates over 70%, TrustMeter reaches perfect identification. As previously, the dropper identification is faster when the clients apply the random-fair strategy compared to the other exchange strategies.

C. Spammers

We analyze the performance of TrustMeter in identifying spammers under the same attack scenarios as in Section IV-B.

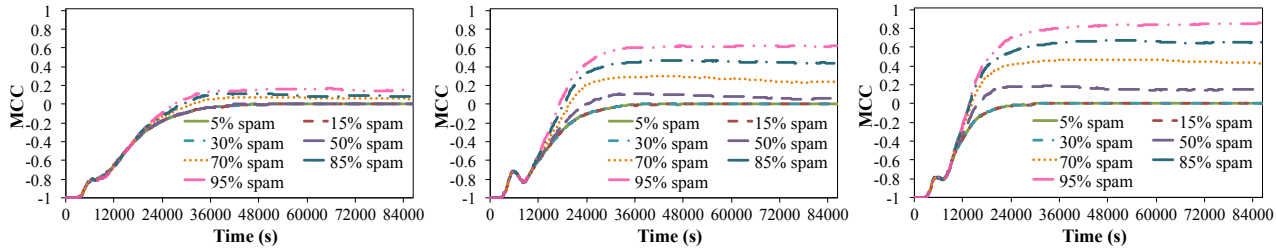
1) Percentage of Spammers: Fig. 4 illustrates the temporal evolution of the MCC for different spammer percentages. For all strategies, the performance of TrustMeter improves with the number of spammers until reaching the best performance for 50% of spammers. The lower the number of spammers, the higher the probability that they exchange spam with honest clients. As a result, the reputation of honest clients decreases, leading to their incorrect classification as *indefinite*. Moreover, the reduction in reputation increases with the number of exchanged spam triplets. Hence, TrustMeter performs better when the clients apply the random-fair strategy for up to 50% of spammers. The probability that spammers exchange spam with other spammers increases with the number of spammers. Their reputation hence diminishes, leading to their correct identification. However, the performance of TrustMeter decreases from 50% of spammers. Since the amount of injected spam increases, the probability that honest clients relay this spam increases. Simultaneously, the impact of the exchange strategy decreases when the number of spammers increases, as the probability to exchange spam increases.

2) Spamming Rate: Fig. 5(a) shows that TrustMeter performs worse in identifying spammers when clients apply the realistic strategy, as clients exchange more triplets than with the other strategies. Therefore, the reputation of honest clients getting spam decreases more rapidly, leading to their incorrect categorization. However, when honest users are identified as *untrusted*, they are temporarily quarantined and can only upload either previously exchanged triplets or their own ones. If the uploaded triplets are correct, clients rapidly regain reputation, thus changing their trust level and being allowed to exchange triplets again. Otherwise, the clients are maintained in quarantine. The performance of TrustMeter improves for the random-unfair strategy due to a lower amount of exchanged triplets and hence, a lower impact on their reputation if the



(a) Realistic strategy (b) Random-unfair strategy (c) Random-fair strategy

Fig. 4. MCC for different percentages of spammers and exchange strategies over time



(a) Realistic strategy (b) Random-unfair strategy (c) Random-fair strategy

Fig. 5. MCC for different spamming rates and exchange strategies over time

triplets are spam. TrustMeter shows the best results for the random-fair strategy with up to a MCC value of 0.85 at the end of the simulation for 100% spamming (cf. Fig. 5(c)).

3) Traffic Overhead: We finally consider 100 clients applying the realistic exchange strategy and observe the distribution of the traffic between the different classes. For sensor readings with a size between 100 bytes to 1 kilobytes, the lists of triplets IDs cause 80% of the total traffic. In comparison, the exchanged triplets and the reputation values only represent 15% and 2% of the traffic, respectively. This is difference is due to the length of the triplet IDs that are currently 32 bytes long using a SHA-256 hash function. The overhead generated by the triplet retransmissions increases with the number of droppers, but remains under 4% of the total traffic even in the worst scenario where all clients drop all triplets. For triplets of larger sizes (around 16 kilobytes [1]), the overhead caused by the lists of triplets IDs represents only 15% of the traffic, whereas the exchange of triplets generates 80% of the traffic.

V. Discussions

We discuss the resilience of TrustMeter against the following attacks based on the threat model introduced in Sec. II-B.

A. Reputation Manipulation

Malicious clients can attempt to alter the computed reputation stored at the server side. The server is, however, protected against unauthorized access using standard cryptographic primitives. As such, this attack cannot be launched. Furthermore, malicious clients can try to report falsified information on behalf of others. Our scheme protects honest clients against this attack by requesting clients to authenticate with the server. Such an attack would only be successful if malicious clients could access the private keys of the targeted clients, which is beyond the scope of our attacker model.

Malicious clients can send falsified information to the server about their exchanges in order to: (1) reduce other clients' reputation, or (2) fraudulently increase their own reputation. However, recall that all exchange partners provide the exchange statistics to the server. As a result, the server can verify that the provided information is the same for both exchange partners. If the figures do not match, the server assumes that one of the exchange partners is providing false information and identify it after several non-matching exchanges.

Another more sophisticated attack is the collusion of malicious clients that may agree on corroborating falsified information to mutually increase their reputation. Since the server maintains a list of exchange partners for each client, an abnormal high-frequency exchange rate between the same clients will not go unnoticed. As a result, the impact of such attacks remains limited. In order to still diminish the effect of this attack, the server can take into account the number of exchange partners in the reputation computation. The more exchange partners, the greater the increase in reputation.

B. Privacy Leaks and Reporting Behaviors

Based on the exchange statistics, the server may, e.g., infer that reported triplets come from either one or the other users if those had only one exchange between two reports. While this insight is not sufficient to definitely link the triplets to their sources, the probability of a correct identification increases if the server is able to timely follow the different exchanges. To reduce this threat, several measures can be applied: (1) multiple exchanges should happen before reporting the triplets, (2) the number of exchanged triplets should be the same for both partners (i.e, using the random-fair strategy), and (3) the users should exchange enough triplets before reporting them. A thorough analysis of the impact of these measures is considered as future work.

C. Replay and Sybil Attacks

Under normal operating conditions, triplet retransmissions are permitted after expiration of the delivery timeout. However, malicious clients can abuse this feature in order to increase their reputation by replaying triplets. The server receiving the same triplets at abnormal rates considers them as spam and utilizes the mechanisms described in Sec. III-B3 to identify the attackers. Moreover, TrustMeter does not prevent attackers from adopting multiple identities. However, attackers need to gain reputation first before effectively launching attacks.

VI. Related Work

Reputation systems have been presented in [7], [9] and [10]. They however concentrate on evaluating the quality of the reported sensor readings. While we integrate such mechanisms to identify spammers, additional mechanisms are necessary to assess the contribution of clients in the collaborative path-hiding mechanism. Therefore, we focus on solutions tailored to delay tolerant and opportunistic networks [11] sharing the more similarities with our system model (see Sec. II-A).

Existing solutions for opportunistic networks can be classified into two categories, depending on the nature of the information used to compute the levels of trust. For example, [12], [13], and [14] leverage social relationships, while [15] and [16] rely on the encounters between users. In particular, users explicitly indicate their degree of trust for other users based on their social relationships and build a web of trust in [13]. However, leveraging social relationships limits the number of exchange partners to a subset of users sharing strong social links and hence, restrains the efficacy of the collaborative path-hiding scheme. Another model proposed in [14] combines information about social relationships and quality of service metrics. However, this model is designed to ensure the delivery of packets between clients using the shortest path. In our approach, the destination of the triplets is always the server and clients are encouraged to exchange triplets with other users before reporting them in order to increase the jumbling and hence, improve the mutual privacy protection of the clients.

Models presented in [15] and [16] determine the trust relationships between users based on their encounters. In [15], users generate a shared secret at each encounter. The shared secret is then used at the next encounter to prove that a previous encounter between the same users already happened. Thus, the trust relationship between users strengthens at each exchange of shared secret. In comparison, the model proposed in [16] analyzes the location of the users over a long period to determine similar behaviors. Users exhibiting similar behavior are assumed to share stronger trust relationships than those showing different behavior. Relying only on encounters is insufficient in our scenario, since malicious users can visit densely populated locations in order to rapidly build strong trust relationships, before launching dropping attacks.

VII. Conclusions

We have presented TrustMeter, a system to assess the behavior of users in collaborative privacy protection mechanisms. We have examined its performance in identifying attackers by means of extensive simulations. The results show that TrustMeter performs well in identifying droppers and spammers despite the multi-hop nature of the scheme. By leveraging TrustMeter, the latency incurred by malicious users is reduced resulting from both their identification and quarantine.

Acknowledgment

This work was supported by CASED (www.cased.de). The paper was written while D. Christin was working at TU Darmstadt. Our thanks go to A. García Bouso for her feedback.

References

[1] D. Christin, A. Reinhardt, S. S. Kanhere, and M. Hollick, "A Survey on Privacy in Mobile Participatory Sensing Applications," *Journal of Systems and Software*, vol. 84, no. 11, 2011.

[2] D. Christin, J. Guillemet, A. Reinhardt, M. Hollick, and S. S. Kanhere, "Privacy-preserving Collaborative Path Hiding for Participatory Sensing Applications," in *Proc. of the 8th IEEE International Conference on Mobile Ad-hoc and Sensor Systems (MASS)*, 2011.

[3] D. Chaum, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms," *Communications of the ACM*, vol. 24, no. 2, 1981.

[4] A. Keränen, J. Ott, and T. Kärkkäinen, "The ONE Simulator for DTN Protocol Evaluation," in *Proc. of the 2nd International Conference on Simulation Tools and Techniques (Simutools)*, 2009.

[5] D. Christin, A. Reinhardt, and M. Hollick, "On the Efficiency of Privacy-Preserving Path Hiding for Mobile Sensing Applications," in *Proc. of the 38th IEEE Conference on Local Computer Networks (LCN)*, 2013,

[6] D. Dolev and A. C. Yao, "On the Security of Public Key Protocols," *IEEE Transactions on Information Theory*, vol. 29, no. 2, 1983.

[7] K. L. Huang, S. S. Kanhere, and W. Hu, "Are you Contributing Trustworthy Data?: The Case for a Reputation System in Participatory Sensing," in *Proc. of the 13th ACM International Conference on Modeling, Analysis, and Simulation of Wireless and Mobile Systems (MSWIM)*, 2010.

[8] D. Christin, D. Rodriguez Pons-Sorolla, S. S. Kanhere, and M. Hollick, "TrustMeter: A Trust Assessment Scheme for Collaborative Privacy Mechanisms in Participatory Sensing Applications—Extended Version," Technische Universität Darmstadt, Online: www.seemoo.de/dl/seemoo/seemoo-tr-2012-02.pdf, Tech. Rep. TR-SEEMOO-2012-02, 2012.

[9] H. Yang, J. Zhang, and P. Roe, "Using Reputation Management in Participatory Sensing for Data Classification," *Procedia Computer Science*, vol. 5, no. 0, pp. 190–197, 2011.

[10] D. Christin, C. Rosskopf, M. Hollick, L. A. Martucci, and S. S. Kanhere, "IncogniSense: An Anonymity-preserving Reputation Framework for Participatory Sensing Applications," in *Proc. of the 10th IEEE International Conference on Pervasive Computing and Communications (PerCom)*, 2012.

[11] L. Lilien, Z. H. Kamal, V. Bhuse, and A. Gupta, "Opportunistic Networks: The Concept and Research Challenges in Privacy and Security," in *Proc. of the International Workshop on Research Challenges in Security and Privacy for Mobile and Wireless Networks (WSPWN)*, 2006.

[12] C. Boldrini, M. Conti, and A. Passarella, "Exploiting Users' Social Relations to Forward Data in Opportunistic Networks: The HiBOp Solution," *Pervasive and Mobile Computing*, vol. 4, no. 5, 2008.

[13] S. Trifunovic, F. Legendre, and C. Anastasiades, "Social Trust in Opportunistic Networks," in *Proc. of the 29th IEEE Conference on Computer Communications (INFOCOM)*, 2010.

[14] I.-R. Chen, F. Bao, M. Chang, and J.-H. Cho, "Trust Management for Encounter-Based Routing in Delay Tolerant Networks," in *Proc. of the IEEE Global Telecommunications Conference (GLOBECOM)*, 2010.

[15] J. Manweiler, R. Scudellari, and L. P. Cox, "SMILE: Encounter-Based Trust for Mobile Social Services," in *Proc. of the 16th ACM Conference on Computer and Communications Security (CCS)*, 2009.

[16] U. Kumar, G. Thakur, and A. Helmy, "PROTECT: Proximity-Based Trust-Advisor using Encounters for Mobile Societies," in *Proc. of the 6th International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2010.