Sensory Data-driven Modeling of Adversaries in Mobile Crowdsensing Platforms

Kyle Quintal, Ertugrul Kara, Murat Simsek, Burak Kantarci, Herna Viktor

***Abstract*—The advent of Mobile crowdsensing (MCS) facilitates the adoption of ubiquitous sensing solutions in smart environments. Despite its benefits, MCS calls for proper security and trust solutions. Various threatening attacks, such as injection attacks, can compromise both the veracity and integrity of crowdsensed data. This work leverages adversarial machine learning to introduce a smart injection attacker model (SINAM) that may be used in the design of security solutions against injection attacks in MCS. SINAM has been validated during an authentic MCS campaign. Unlike most random data injection models, SINAM monitors data traffic in an online-learning manner, successfully injecting malicious data across multiple victims with near-perfect accuracy rates of 99%. SINAM uses accomplices within the sensing campaign to predict accurate injections based on both behavioral analysis and context similarities.**

***Index Terms*—Artificial Intelligence, Cybersecurity, Machine Learning, Mobile Crowdsensing, Internet of Things, Mobile computing**

I. Introduction

Mobile Crowdsensing (MCS) empowers smart devices in providing non-dedicated and ubiquitous sensing tasks in the Internet of things (IoT) Era [1], [2]. As a subsystem of an IoT environment, MCS campaigns acquire sensed data that are heterogeneous and large in volume. Widely referred to as the big data phenomenon, the "big sensed data" [3] is a key enabler for various application in smart cities and spaces [4], [5]. Live MCS campaigns for large scale sensing [6] and ongoing research on many smart applications point out the potential of MCS to become an integral component of IoT-based services [7], [8].

The ubiquity of sensing devices, alongside their participatory and opportunistic sensing service provisioning all introduce the challenge of trustworthiness assurance against malicious participants [9]. Related work reports that malicious and untrustworthy participants in a crowdsensing network can leverage the system for either personal gain or disinformation at the crowdsensing servers / platform [10]–[12]. There are many threats sourced by malicious participants, such as information tampering, privacy breaches, and service denials just to name a few.

These threats differ in many aspects, including frequency, complexity, and potential damage, the modeling of every single type remains an open issue against wide adoption of MCS systems. Based on these observations, resilient attack measures should be integrated with MCS systems [13]. However, the security design of MCS systems should be verified under realistic and valid attack models, which still requires in-depth study.

Adversarial machine learning uses machine learning principles to produce malicious inputs to mislead a system [14]. Our paper leverages this concept to present a smart injection attacker model (SINAM) to help future MCS contributions prevent malicious data manipulations, specifically regarding data injection. Three fundamental assumptions of SINAM are as follows:*i) Victims:* are MCS users which are monitored for potential injections, *ii) Accomplices:* are attackers within the MCS campaign that collaborate and exchange information to improve undetected injection rates, *iii) Similar settings:* denotes a shared context between victims and accomplices, in order for intelligent injections to occur. If victims and accomplices are in two completely distinct environments, injecting malicious data based on the behavior of accomplices is not as feasible.

Intuitively, injection attacks continuously insert malicious data in continuous streams of incoming data. Our case study involves injections in such a live sensing campaign. We propose SINAM, a smart injection attack model, which tracks behavioral similarities and uses a novel constant called the *tampering potential* to make intelligent injection decisions. As data is processed in batches, SINAM uses the behavior of accomplices within the MCS campaign to create undetectable injections, which are inserted into the victim's data streams. This is accomplished by considering similar behavior between accomplices and the selected victim and identifying which features are vulnerable for injection. Feature injections are produced using the proposed tampering potential, which defines the ratio at which victim and accomplice behaviors are used for injection. SINAM is successful in deceiving our MCS campaign by achieving up to 100% undetected injection rates under certain parameters.

The rest of the paper is as follows: Section II discusses the related work on the identification and prevention of attacks in MCS. Section III presents the details and formulation of SINAM. Section IV presents and discusses the numerical results regarding our behavioral analysis. Finally, the paper is concluded in Section V, where future directions are presented.

II. Related work

Among many challenges in MCS, improving user participation and defining effective incentives is a key challenge [15]. Maximum utility or profit is the main objective of both

The authors are with the School of Electrical Engineering and Computer Science at the University of Ottawa, Ottawa, ON, K1N 6N5, Canada. E-mail: {kquin039, ekara044, burak.kantarci, murat.simsek, hviktor}@uottawa.ca

978-1-7281-0962-6/19/$31.00 ©2019 IEEE

MCS platforms and participants [16]. Indeed, by exploiting auction and game theories, researchers have developed various incentive mechanisms for MCS users [17]. However, incentives aiming at maximum utility must be coupled with security, trustworthiness and robustness solutions against intrusive behavior of malicious users. For instance, the ability to assign multiple sensing tasks to individuals allows parallel sensing tasks to be undertaken by the same user, thus minimizing the required threshold for triggering sensing tasks [18]. Furthermore, the efficient use of sensors introduces vulnerabilities, especially if participant rewarding is attributed to task completion. Continuous authentication is a strong tool to cope with these vulnerabilities [19]. Continuously authenticating participants could protect location and data privacy during sensing tasks [20] while also respecting their anonymity [21]. Ensuring the authenticity of participants in an MCS system can also help to prevent Sybil attacks [22], as well as to identify Denial-of-Service (DoS) attacks by leveraging game theory methods [23].

MCS systems need robust security measures with the ability to prevent breaches, protect user privacy and system integrity. To verify security solutions, various machine learning techniques are being introduced to model different sensing attacks. One such example may be seen in [24], where Stackelberg game theory is used alongside deep reinforcement learning. The authors in [25] have also investigated the feasibility of AI-based detection of clogging attacks in MCS environments. The protection of a user's location and data privacy have also been achieved in a real-world experiment [20], helping reduce the gap towards industrial implementations. Based on the above observations, this study builds on similar motivation as the social network authentication studies performed on real users in [26], [27], but focuses on modeling realistic attacks for real-time "big sensed data" acquisition campaigns on a physical crowdsensing platform.

III. System Model and Methodology

SINAM monitors streams of incoming data and combines them into batches, which are injected as follows: *i) Monitoring:* Monitor the selected victim's data until N data points are obtained. These create the victim's batch, B_v. *ii)Behavior Analysis:* First, each index of the victim's batch is associated with an accomplice behavior through a distance-based selection process. Second, every behavior is injected through an accomplice influence-aware process (at a rate of α) by ensuring deviations occur within each of the victim's features. *iii) Injection:* Finally, behavior injections are combined using their corresponding (accomplice's) weights to create the final injection.

These three above steps are repeated once another N data points create a new batch. A minimalist illustration of the system under consideration is presented in Fig. 1. Attacks on multiple victims require multiple replicas of SINAM, each model only injecting on a single victim. The notation for formulations is presented in TABLE I.

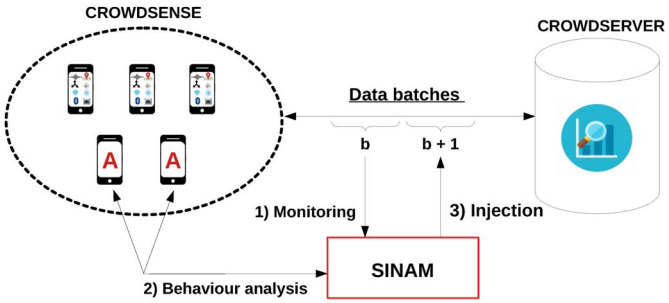

Fig. 1. 3-step model: 1) Obtaining data for attack, 2) Conducting analysis and computation of injection, 3) Injection into next batch of victim's data

TABLE I
Notation for Adversarial Model

Notation	Description
N	Batch size
n	index of batch
F	Set of features
f	instance of F
A	Set of accomplices
k	instance of A
$\mathcal{B}_v$	Batch of victim's data
$\mathcal{B}_{a_k}$	Batch of accomplice k's data
$\mathcal{B}^n_{a_k}$	index n of Ba_k
$\mathcal{B}^n_v$	index n of B_v
v^n_f	feature f at index n of Bv
$a^n_{k,f}$	feature f at index n of Ba_k
α	tampering potential
i_k	closest behavior to B^i_v found in $B^i_{a_k}$
W_k	Weight given to accomplice k
φ_k	Tampered injections of accomplice k
Φ_k	Selected injections of accomplice k
W_f	Combined injections of feature f
$\Psi(B^{N+1}_v)$	Injection inserted in succeeding batch

A. Monitoring

The initial step is obtaining N data points from the targeted victim. In comparison to online learning terminology, streams of information are henceforth called data points. Each data point has F features, as seen in (1), which represents a single instantaneous behavior of the victim.

$$B^n_v = \begin{bmatrix} v^n_{f1} & v^n_{f2} & \dots & v^n_{fF} \end{bmatrix}^T \tag{1}$$

Once N collected instantaneous behaviors have been collected, these are combined to create our victim's batch, seen as the victim's behavior patterns, or (2). Each instantaneous behavior represents an index of the batch, as shown in (3).

$$B_v = \begin{bmatrix} B^1_v & \dots & B^n_v & \dots & B^N_v \end{bmatrix}_{(F\times N)} \tag{2}$$

$$B_v = \begin{bmatrix} \begin{bmatrix} v^1_{f1} \\ v^1_{f2} \\ \vdots \\ v^1_{fF} \end{bmatrix} & \dots & \begin{bmatrix} v^n_{f1} \\ v^n_{f2} \\ \vdots \\ v^n_{fF} \end{bmatrix} & \dots & \begin{bmatrix} v^N_{f1} \\ v^N_{f2} \\ \vdots \\ v^N_{fF} \end{bmatrix} \end{bmatrix} \tag{3}$$

The influence on the selected size of N is discussed in a later section.

B. Behavior Analysis

This step is composed of two parts: first, selecting the closest accomplice behaviors based on our distance function and second, injecting those closest behaviors based on the *tampering potential*.

1) Behavior Selection: Every accomplice's batch is compared with the victim's, index-by-index (behavior-by-behavior). Each index, therefore, contains the closest accomplice behavior of that index, denoted as i_k.

$$i_k = \arg\min_i(||\mathcal{B}_v^i - \mathcal{B}_{a_k}^i||);\ i = 1, .., N,\ k = 1, ..., A \quad (4)$$

Each closest behavior for all accomplices is denoted by $[\mathcal{B}_v^{i_k}, \mathcal{B}_{a_k}^{i_k}]$ for all accomplices. The next step is the normalization of the closest behaviors depending on behavior closeness. Closeness is denoted as a smaller Euclidean distance to the victim's index, as seen in (5).

$$W_k = \frac{(||\mathcal{B}_v^{i_k} - \mathcal{B}_{a_k}^{i_k}||)^{-1}}{\sum_{j=1}^{A}(||\mathcal{B}_v^{i_j} - \mathcal{B}_{a_j}^{i_j}||)^{-1}} \quad \forall k \in A \quad (5)$$

2) Behavior Tampering: Here, the behavior is injected in a tampering potential-aware manner, meaning the lower the tampering potential, the less an accomplice will inject its behavior. The ratio of injected behavior is defined by the tampering potential α. A higher injection ratio causes additional behavior injection but is more easily detected. This process is formulated in (6).

$$\varphi_k = (1 - \alpha) \cdot B_{a_k}^{i_k} + \alpha \cdot B_v^{i_k},\ \ k = 1, 2, ..., A \quad (6)$$

When a specific feature within the victim's batch does not contain temporal deviations, it is not injected with behavior. For example, if a victim's location remains unchanged within the entire batch, any behavior injection in such a stable feature would be much easier to detect. Thus, injections only occur when the standard deviation (7) of a feature over the batch is true. On the contrary, stable features (false δ) are duplicated into the injection, (8)

$$\delta = \begin{bmatrix} \delta_1 & \delta_2 & \dots & \delta_F \end{bmatrix}^T \quad (7)$$

$$\Phi_k = \begin{bmatrix} B_{v1}^{i_k} \cdot \overline{\delta_1} + \varphi_{k1} \cdot \delta_1 \\ \vdots \\ B_{vF}^{i_k} \cdot \overline{\delta_F} + \varphi_{kF} \cdot \delta_F \end{bmatrix}_{(F \times 1)}, k = 1, 2 \dots, A \quad (8)$$

The feature tampering process is illustrated in Fig.2, demonstrating the influence of the tampering potential(α) on the final injection.

C. Injection

The behavior closeness normalization defined in (5) is the weight given to each tampered feature, defined in (8). This weighted sum creates the final injection vector, or Ψ.

$$\Psi(B_v^{N+1}) = \begin{bmatrix} \sum_{k=1}^{A} \Phi_{k_1} \cdot W_k \\ \vdots \\ \sum_{k=1}^{A} \Phi_{k_F} \cdot W_k \end{bmatrix}_{(F \times 1)} \quad (9)$$

Once computed, the injection is inserted in the succeeding victim's data batch (N+1) and the process is repeated, by attacking another (or potentially the same) victim. All previous injections are overlooked by SINAM in succeeding behavioral analysis' since they do not represent true victim behavior.

D. Random Attack Model

Our proposed method, SINAM, conducts the three above steps(monitoring, behavioral analysis, and injection), but we also consider a randomized attacker model as a baseline model. This baseline model skips the accomplice selection phase, instead randomly selecting accomplice behaviors for injections. Thus, SINAM selects based on behavior similarities and the Random Attack Model makes a random selection.

IV. Performance Evaluation

This section explains the sensing campaign and characteristics of the collected data, alongside feature selection and behavioral clustering. The undetected injection rates are compared between SINAM and randomized attacker to demonstrate the effectiveness of SINAM.

A. Sensing Campaign and Dataset

To evaluate the effectiveness of SINAM, a sensing campaign was conducted on the main campus of the University of Ottawa with 9 volunteering participants. Data extraction occurred from an installed Android application, which extracted 30 different sensors (including multidimensional values) at a set interval of 20 seconds, over three weeks. Since the crowd-sensing application allowed participants to disable unwanted feature collection, a few features were quite sparse. Such features, alongside uncommon sensors which caused similar data sparsity, were removed for fair behavior representations.

The remaining features were analyzed using principal component analysis to identify the highly-contributing features of behavior representation. Some obvious contributions emerged (lat and lon), but others were surprising. Two notable contributions were the orientation azimuth and proximity. These features were among the highest contributions across all users. The hypothesizes reasoning is as follows: humans now tend to place mobile devices on specific flat surfaces (office desk for instance). Each workstation itself has a specific azimuth, despite their slight variations. As these minor, but yet significant, differences are combined with proximity (being face-up or face-down), they could thus mimic phone placement habits of individuals, although further confirmation is required to

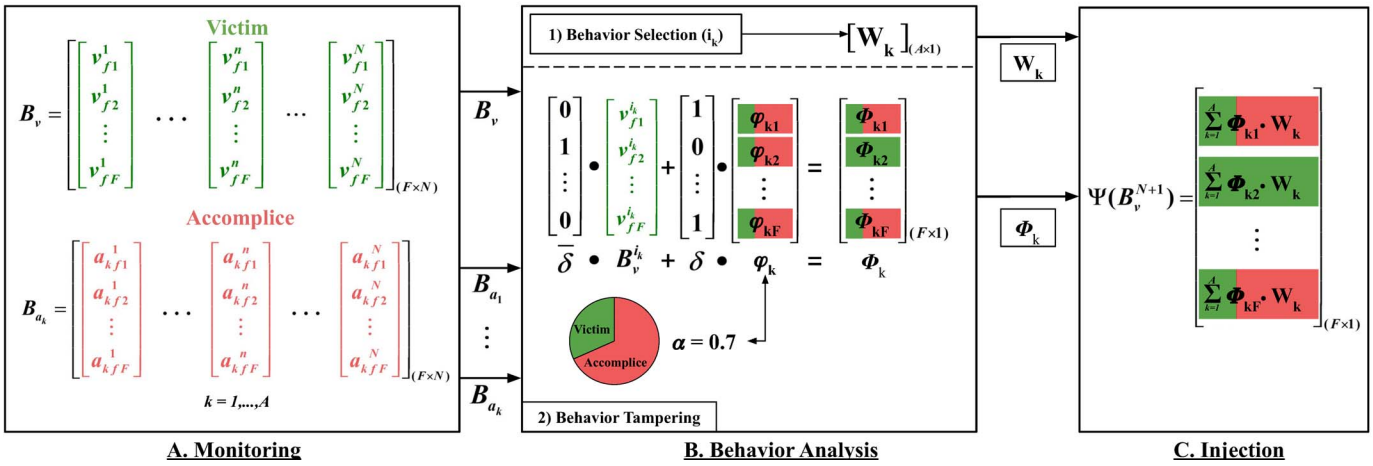

Fig. 2. Influence of tampering potential on injected behavior

confirm this hypothesis. The 15 most contributing features were selected for behavioral analysis. These are as follows: accelerometer x, accelerometer y, accelerometer z, orientation azimuth, orientation roll, orientation pitch, latitude, longitude, battery level, battery consumption, weekday, the hour of the day, microphone intensity, proximity, and WiFi usage.

B. Behavioral clustering

The users were split into two groups: the *'victims'* (Users 1-2-3-7-8) and the *'accomplices'* (Users 4-5-6-9). The victims were selected since they covered over 90% of the dataset and had complete behavior profiles (most active users in the campaign). The remaining users became accomplices and contained incomplete behavior profiles (less active during the collection campaign). Behavior clustering was conducted on victims using DBSCAN to distinguish our victim's behaviors.

TABLE II
BEHAVIORAL CLUSTERING WITH DBSCAN

Users	Total Data Points	Number of clusters	Accuracy
User 1	8551	4	0.9979
User 2	12842	5	0.9347
User 3	22680	2	0.9985
User 7	12009	10	0.9851
User 8	6386	5	0.9432

DBSCAN performs especially well on spatial clustering [28], the equivalency of distinctive behavioral patterns in our context. DBSCAN requires two key parameters, epsilon(ϵ) and the minimum samples to form a cluster. The former denotes the maximum distance between two points for them to be considered as belonging to a single cluster. These parameters have been selected empirically, and we have found that ϵ = 0.4, and the 50 minimum samples lead to reasonable accuracy and behavior clusters (i.e. distinct behavioral patterns). Indeed, behavior pattern distinctions are not identical across all victims, but behavior identification for individual users remains high, as seen in Table II. The accuracy levels were obtained by using 25% of the victim's data as validation at the end of the sensing campaign.

C. Detecting Behavior Injections

The effectiveness of both models is represented with the undetected injection rate, which denotes the rate at which injections are classified as the victim's behavior. The impact of different *tampering potentials* and batch sizes are also presented. A batch size represents the quantity of data (behavior patterns) that are analyzed before injection decisions are made by attackers. The smart attacker model (SINAM) is compared with the simplistic random attacker model, which we denote as *Random Attack Model*. It is worth noting that the baseline Random Attack Model, as well, inherently leverages intelligence to some extent, as it performs the feature tampering step seen in Section III-B2. Different values of *tampering potential* are presented to demonstrate its impact on undetected injection rates.

As DBSCAN formed distinct behavior clusters for each victim, an event of data injection is assumed undetected if the injected data point is clustered in one of the victim's behavioral clusters. The success metric is defined as the undetected injection rate.

Our experiment was conducted using batch sizes of 50,100 and 200, each with variations in *tampering potentials* being 0, 0.5 or 1. Both the SINAM and Random Attack Model are compared in Table III. As depicted a methodological overview, if a victim's feature does not change within a monitored batch, both attack models refrained from injecting accomplice behaviors into that particular feature, seen in Fig.2.

This approach was taken as victims sometimes demonstrated very little fluctuations in data. For instance, stationary users constantly send the same latitude and longitude values until they move around again. Thus producing a single injection in such stable features would result in immediate injection detection. Alternatively, when there are fluctuations within a feature of a victim's batch, the *tampering potential* kicks in. A *tampering potential* of 1 represents full accomplice injections and a *tampering potential* of 0 causes both attack models to

TABLE III
UNDETECTION RATES ACROSS DIFFERENT CONFIGURATIONS

	Attackers	BatchSize=50			BatchSize=100			BatchSize=200		
		α:0.0	α:0.5	α:1.0	α:0.0	α:0.5	α:1.0	α:0.0	α:0.5	α:1.0
User 1	SINAM	1.0000	1.0000	0.7143	1.0000	1.0000	0.5273	1.0000	1.0000	0.3500
	Random	1.0000	0.7262	0.0893	1.0000	0.6848	0.1273	1.0000	0.7000	0.1500
User 2	SINAM	0.9451	0.8537	0.7053	0.9489	0.8043	0.7404	0.9619	0.7619	0.7238
	Random	0.9451	0.4675	0.0711	0.9489	0.3617	0.0936	0.9619	0.4667	0.1048
User 3	SINAM	1.0000	1.0000	0.2779	1.0000	1.0000	0.0000	1.0000	0.9954	0.0000
	Random	1.0000	0.3114	0.0011	1.0000	0.3386	0.0045	1.0000	0.4537	0.0000
User 7	SINAM	0.9893	0.9850	0.9079	0.9956	0.9780	0.8943	0.9906	0.9340	0.7830
	Random	0.9893	0.8158	0.4304	0.9956	0.7137	0.3833	0.9906	0.7547	0.4717
User 8	SINAM	0.9544	0.9087	0.9627	0.9643	0.8840	0.8929	1.0000	0.7959	0.7347
	Random	0.9544	0.7635	0.5062	0.9643	0.7054	0.4732	1.0000	0.7143	0.3469

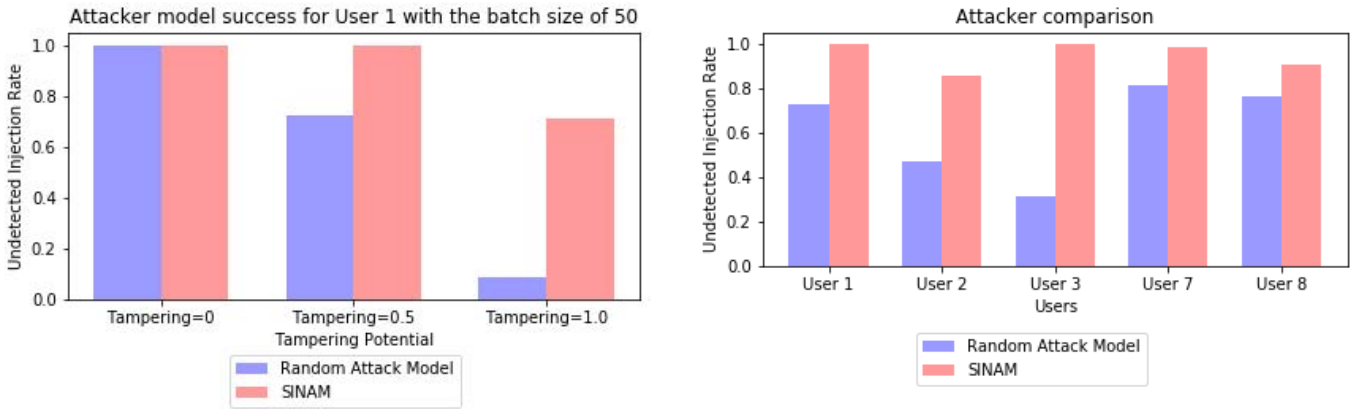

Fig. 3. Impact of the tampering potential for User 1 (left), and undetected injection rate under Randomized Attacker and SINAM with a batch size of 50 and tampering potential of 0.5 (right)

use all of the victim's original features, rendering the injection a copy of the victim's behavior but not a malicious injection.

The results support the effectiveness of SINAM compared to the Random Attack Model. This gap in performance only increases as the *tampering potential* increases, causing a higher ratio of accomplice behavior to be in the injections. The Random Attack Model is penalized as the randomness in behavior selection from the Random Attack Model causes more frequent anomaly detection by DBSCAN, since the closeness (compatibility) between accomplice and victim behavior is not considered systematically. SINAM's advantage in the accomplice selection phase (III-B1) leads injections to have always had a higher undetected injection rate, regardless of batch size and *tampering potential*.

Another remark is seen as the batch sizes decrease, the undetected rates increase. These batch sizes cause high frequencies of similar behavioral data, creating a higher volume of victim-accomplice comparisons. With more comparisons, there is a higher potential to find similar behaviors in the resulting set of behavioral patterns (set S in formulations). This set then becomes less diluted, thus having more sporadic behavior patterns, which are more vulnerable to injection. Having too small batch sizes would cause behavior patterns to become behavioral noise, as behavior is defined as a series of actions to events, but requires sufficient actions for any fruitful behavioral extraction.

Fig. 3a demonstrates different *tampering potential* values and how they influence undetected injection rates for both the Random Attack Model(blue bars) and SINAM (red bars). As shown in the figure, the tampering potential dramatically affects the Random Attack Model, as it is the only property keeping that model from being purely random. A tampering potential of 0 signifies injections are purely based on the previously repeated victim behaviors, but tampering potential of 1 indicates full behavior injections taken from the accomplices. Hence, increasing the tampering potential decreases the undetected injection rates for the Random Attacker Model more drastically compared to SINAM, who barely receives

a drop at 0.5. With a tampering potential of 1, undetected injection rate drops to near zero for the Random Attack Model, but the SINAM model remains acceptable around 70%. This observation is only on a single victim (user1) with a batch size of 50.

Fig. 3b presents an overall comparison of the Random Attack Model and SINAM with a batch size of 50 and a *tampering potential* of 0.5. As seen in the TABLE III, behavioral patterns of a certain users (user 3 for example) had more clear cutoffs. As expected, the more behavioral clusters a victim had (see TABLE II), the better both models performed. However, for most users and majority of the population, SINAM was able to inject data with at least 90% undetection when using a *tampering potential* of *0.5*.

V. Conclusion

This work introduces Smart Injection Attacker Model (SINAM) an attacker model that may potentially be used in the development of security solutions for Mobile Crowdsensing (MCS) systems. Both SINAM and Random Attack Model (i.e. baseline) rely on the collaborations with accomplices within an MCS campaign to successfully inject behavior patterns into a victim's data batches. Experimental results confirm the behavior injections of SINAM were adequately effective, with undetected injections rates ranging from 85-100% when using a batch size of 50 and tampering potential of 0.5, one of many potentially successful configurations. The introduced model may be applied to test authentication or identification methods, alongside other applications, that would benefit from an intelligent injection model within MCS campaigns.

We are currently extending this model to identify the most vulnerable victims to have dynamically optimized values of the tampering potential (α). This is also applicable to the selection of batch sizes and feature selection, which could benefit from further granularity to achieve optimal selections. Furthermore, distance metrics other than Euclidean distance could potentially be investigated in future work.

Acknowledgement

This study has been supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada under the DISCOVERY program.

References

[1] J. Liu, H. Shen, H. S. Narman, W. Chung, and Z. Lin, "A survey of mobile crowdsensing techniques: A critical component for the internet of things," *ACM Trans. on Cyber-Physical Systems*, vol. 2, no. 3, p. 18, 2018.

[2] H. Habibzadeh, Z. Qin, T. Soyata, and B. Kantarci, "Large-scale distributed dedicated-and non-dedicated smart city sensing systems," *IEEE Sensors Journal*, vol. 17, no. 23, pp. 7649–7658, 2017.

[3] S. M. Oteafy and H. S. Hassanein, "Big sensed data: Evolution, challenges, and a progressive framework," *IEEE Communications Magazine*, vol. 56, no. 7, pp. 108–114, 2018.

[4] Y. Sun, H. Song, A. J. Jara, and R. Bie, "Internet of things and big data analytics for smart and connected communities," *IEEE access*, vol. 4, pp. 766–773, 2016.

[5] M. Pouryazdan and B. Kantarci, "The smart citizen factor in trustworthy smart city crowdsensing," *IT Professional*, vol. 18/4, pp. 26–33, 2016.

[6] G. Cardone, A. Corradi, L. Foschini, and R. Ianniello, "Participact: A large-scale crowdsensing platform," *IEEE Trans. on Emerging Topics in Computing*, vol. 4, no. 1, pp. 21–32, 2015.

[7] A. Capponi, C. Fiandrino, B. Kantarci, L. Foschini, D. Kliazovich, and P. Bouvry, "A survey on mobile crowdsensing systems: Challenges, solutions, and opportunities," *IEEE Communications Surveys Tutorials*, vol. 21, no. 3, pp. 2419–2465, Third Quarter 2019.

[8] G. Cardone, L. Foschini, P. Bellavista, A. Corradi, C. Borcea, M. Talasila, and R. Curtmola, "Fostering participaction in smart cities: a geo-social crowdsensing platform," *IEEE Communications Magazine*, vol. 51, no. 6, pp. 112–119, 2013.

[9] 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 of Things Journal*, vol. 5, no. 4, pp. 2971–2992, 2017.

[10] B. Kantarci and H. T. Mouftah, "Trustworthy sensing for public safety in cloud-centric internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 4, pp. 360–368, 2014.

[11] J. Ni, A. Zhang, X. Lin, and X. S. Shen, "Security, privacy, and fairness in fog-based vehicular crowdsensing," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 146–152, 2017.

[12] M. Pouryazdan, C. Fiandrino, B. Kantarci, T. Soyata, D. Kliazovich, and P. Bouvry, "Intelligent gaming for mobile crowd-sensing participants to acquire trustworthy big data in the internet of things," *IEEE Access*, vol. 5, pp. 22 209–22 223, 2017.

[13] M. Pajic, J. Weimer, N. Bezzo, O. Sokolsky, G. J. Pappas, and I. Lee, "Design and implementation of attack-resilient cyberphysical systems: With a focus on attack-resilient state estimators," *IEEE Control Systems Magazine*, vol. 37, no. 2, pp. 66–81, 2017.

[14] P. McDaniel, N. Papernot, and Z. B. Celik, "Machine learning in adversarial settings," *IEEE Security & Privacy*, vol. 14/3, pp. 68–72, 2016.

[15] X. Zhang, Z. Yang, W. Sun, Y. Liu, S. Tang, K. Xing, and X. Mao, "Incentives for mobile crowd sensing: A survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 54–67, 2015.

[16] T. Luo, H.-P. Tan, and L. Xia, "Profit-maximizing incentive for participatory sensing," in *IEEE INFOCOM 2014-IEEE Conference on Computer Communications*. IEEE, 2014, pp. 127–135.

[17] D. Yang, G. Xue, X. Fang, and J. Tang, "Incentive mechanisms for crowdsensing: Crowdsourcing with smartphones," *IEEE/ACM Trans. on Networking*, vol. 24, no. 3, pp. 1732–1744, June 2016.

[18] B. Guo, Y. Liu, W. Wu, Z. Yu, and Q. Han, "Activecrowd: A framework for optimized multitask allocation in mobile crowdsensing systems," *IEEE Trans. on Human-Machine Systems*, vol. 47/3, pp. 392–403, 2016.

[19] L. Fridman, S. Weber, R. Greenstadt, and M. Kam, "Active authentication on mobile devices via stylometry, application usage, web browsing, and gps location," *IEEE Systems Journal*, vol. 11/2, pp. 513–521, 2016.

[20] Q. Ma, S. Zhang, T. Zhu, K. Liu, L. Zhang, W. He, and Y. Liu, "Plp: Protecting location privacy against correlation analyze attack in crowdsensing," *IEEE Trans. on Mobile Computing*, vol. 16, no. 9, pp. 2588–2598, 2016.

[21] Y. Shi, Q. Zhang, J. Liang, Z. He, and H. Fan, "Obfuscatable anonymous authentication scheme for mobile crowd sensing," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2918–2929, Sep. 2019.

[22] L. Jiang, X. Niu, J. Xu, Y. Wang, Y. Wu, and L. Xu, "Time-sensitive and sybil-proof incentive mechanisms for mobile crowdsensing via social network," *IEEE Access*, vol. 6, pp. 48 156–48 168, 2018.

[23] N. Ruan, L. Gao, H. Zhu, W. Jia, X. Li, and Q. Hu, "Toward optimal dos-resistant authentication in crowdsensing networks via evolutionary game," in *2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2016, pp. 364–373.

[24] L. Xiao, Y. Li, G. Han, H. Dai, and H. V. Poor, "A secure mobile crowdsensing game with deep reinforcement learning," *IEEE Trans. on Information Forensics and Security*, vol. 13, no. 1, pp. 35–47, Jan. 2018.

[25] Y. Zhang and B. Kantarci, "Ai-based security design of mobile crowdsensing systems: Review, challenges and case studies," in *2019 IEEE International Conference on Service-Oriented System Engineering (SOSE)*. IEEE, 2019, pp. 17–1709.

[26] F. Anjomshoa, M. Aloqaily, B. Kantarci, M. Erol-Kantarci, and S. Schuckers, "Social behaviometrics for personalized devices in the internet of things era," *IEEE Access*, vol. 5, pp. 12 199–12 213, 2017.

[27] Z. I. Rauen, F. Anjomshoa, and B. Kantarci, "Gesture and sociability-based continuous authentication on smart mobile devices," in *Proc. of the 16th ACM Intl. Symp. on Mobility Management and Wireless Access*, 2018, pp. 51–58.

[28] T. Ali, S. Asghar, and N. A. Sajid, "Critical analysis of dbscan variations," in *2010 International Conference on Information and Emerging Technologies*. IEEE, 2010, pp. 1–6.