



Deposited via The University of Leeds.

White Rose Research Online URL for this paper:

<https://eprints.whiterose.ac.uk/id/eprint/213511/>

Version: Accepted Version

Proceedings Paper:

Boussad, Y., Yang, Y., Tomlinson, A. et al. (2024) McMatcher: A symbolic representation for matching random BLE MAC addresses. In: 2024 IEEE International Conference on Consumer Electronics (ICCE). 2024 IEEE International Conference on Consumer Electronics (ICCE), 06-08 Jan 2024, Las Vegas, NV, USA. IEEE. ISBN: 979-8-3503-2414-3. ISSN: 2158-3994. EISSN: 2158-4001.

<https://doi.org/10.1109/icce59016.2024.10444395>

© 2024 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Reuse

Items deposited in White Rose Research Online are protected by copyright, with all rights reserved unless indicated otherwise. They may be downloaded and/or printed for private study, or other acts as permitted by national copyright laws. The publisher or other rights holders may allow further reproduction and re-use of the full text version. This is indicated by the licence information on the White Rose Research Online record for the item.

Takedown

If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

McMatcher: A symbolic representation for matching random BLE MAC addresses

Yanis Boussad

Institute for Transport Studies
University of Leeds
Leeds, UK
y.boussad@leeds.ac.uk

Yuanxuan Yang

Institute for Transport Studies
University of Leeds
Leeds, UK
y.yang6@leeds.ac.uk

Andrew Tomlinson

Institute for Transport Studies
University of Leeds
Leeds, UK
a.tomlinson@leeds.ac.uk

Susan Grant-Muller

Institute for Transport Studies
University of Leeds
Leeds, UK
s.m.grant-muller@its.leeds.ac.uk

Abstract—Bluetooth Low Energy (BLE) is a widely used wireless technology which offers a wide range of applications. However, the introduction of MAC address randomization to preserve the users’ privacy makes it more challenging to leverage all its potential. In this paper, we present McMatcher, a privacy-preserving, novel methodology for matching random MAC addresses from BLE devices. McMatcher uses a symbolic representation of the RSSI time series to build characterizing vectors, embedding both the temporal as well as the signal strength (RSSI) properties of the BLE signal. Our methodology achieves 100% accuracy in matching 92 MAC addresses from 16 smartphones, in a dataset containing 332 MAC addresses in total. As opposed to previous works, our methodology does not require any model training, and relies only on the RSSI measurements. The computational simplicity of McMatcher allows matching MAC addresses in realtime, taking only 230ms for a set of 18 MAC addresses.

Index Terms—Bluetooth Low Energy, MAC address randomization, RSSI, IoT, security and privacy, signal processing,

I. INTRODUCTION

Bluetooth technology, especially Bluetooth Low Energy (BLE), is one of the most used wireless technology, found in a wide variety of equipment such as smartphones and smartwatches [1]. This makes it a promising communication standard for the Internet of Things (IoT) and smart cities. In 2022, there were 4.9 billion Bluetooth-enabled devices shipped worldwide, and it’s expected to reach 7.6 billion by 2027 [2]. Bluetooth-enabled devices contain a hardware interface that implements the Bluetooth protocol for receiving and transmitting Bluetooth signals. This interface has a unique hardware identifier called the Medium Access Control (MAC) address. This identifier is shared between devices when communicating, but also when broadcasting and *advertising* their presence for other devices to discover it [1]. However, such identifiers can be used to track individuals, which represents a risk to their privacy [3, 4]. To avoid tracking users using the MAC address and preserve their privacy, modern BLE devices change their MAC address by generating new random one periodically. This is called MAC address randomization [4],

introduced in BLE standard version 4.0[5]. However, this desire for privacy achieved through MAC address randomisation causes challenges for technological solutions based on BLE technology [6], and most recently, for contact tracing during the Covid-19 pandemic [7, 8].

Related works

Many works have already been undertaken to try to overcome BLE MAC randomisation [5, 9, 10, 11, 12, 13]. Some approaches relied on vulnerabilities in the BLE protocol and exploited them to identify the device generating the random addresses, while others relied on signal characteristics such as signal strength and advertisement intervals for fingerprinting devices. Jouans et al. [11] proposed a strategy for associating randomized MAC addresses of the same device. The authors use the delay after a MAC address has disappeared to select the new MAC address to associate it with. This delay is defined to be the time a device takes to swap its MAC address, and is calibrated as a multiple of the advertising interval of the device T_{int} . The authors then solve a linear assignment problem and using T_{int} as a characterizing identifier of each device. The difference between the swap delays of the MAC addresses is used as a distance metric. However, T_{int} , ranging between 20 milliseconds and 10.24 seconds, requires a robust and fast scanning sniffer in order to measure T_{int} correctly, which limits this technique to only a set of specialized sniffers. Another work that used signal strength information as well as the temporal characteristics of the BLE signals is presented in [12]. Akiyama et al. use the time difference between the disappearing MAC address and the appearance of the new ones and computes the average RSSI for each MAC address. A new MAC address is considered as a candidate for the same device if the time difference is below a certain threshold and the difference between their average RSSI is also below a predefined threshold R . A normalized distance is then computed for each candidate based on time and average RSSI difference, and the candidate with the smallest distance is selected. However, relying on the average RSSI may be not enough given the fluctuation of BLE signals in realistic scenarios. Gagnon et al. [13] presented an RSSI-based fingerprinting of BLE devices by training a machine learning model to match signals from the same device. First,

This work was supported by the Lifeband project TS/V015788/1, funded by Innovate UK, and TRACK project EPv032658/1 funded by Research Council UK, in collaboration with IMPLI and DiscovrAnalytics, members of the Lifeband consortium.

a feature vector (profile) is built for each transmitter from the normalized histogram of its RSSI values. A machine learning model is then trained to match profiles from the same device. However, this method is not suitable for realtime matching as it requires model training.

In this work, we present a novel methodology McMatcher, that relies solely on the BLE RSSI signal properties to match random MAC addresses generated by the same device. The main contributions of this paper are the following.

- **RSSI-based:** Our methodology embeds both the temporal and the signal strength properties of an RSSI time-series which can be easily and passively collected by a sniffer for matching randomized MAC addresses.
- **Matching quality:** When matching different MAC addresses data, McMatcher also computes the matching quality using cosine similarity.
- **No model training required:** McMatcher uses Cosine similarity as a metric for matching, and does not require training a machine learning model, making it computationally simple, and suitable for realtime matching.

This paper is organized as follows. In section II we describe the details of McMatcher. We present the evaluation setup in section III, then we present the results and discuss them in section IV. We conclude this paper in section V.

II. METHODOLOGY

In this section, we present our methodology for linking random MAC addresses of the same BLE device. Suppose that we have a set of BLE devices. Each device broadcasts BLE beacons that can be collected using a measurement tool. Measured beacons typically include the MAC address of the transmitting device and the signal strength (RSSI), expressed in dBm. A sequence of timestamped values received from the same MAC address M forms a time-series \mathcal{T}^M (Equation 1).

$$\mathcal{T}^M := \{(t_i^M, R_i^M) : i = 1, 2, 3, \dots\} \quad (1)$$

The collection of many time-series from the set of Bluetooth devices forms a dataset $\mathcal{D} := \{\mathcal{T}^{M1}, \mathcal{T}^{M2}, \mathcal{T}^{M3} \dots\}$. Devices may use MAC address randomization to change their MAC address during data collection. The goal is to match and join the set of time-series in \mathcal{D} that are broadcast by the same physical device.

A. From RSSI time-series to SAX vectors

We transform each time-series \mathcal{T}_M using Symbolic Aggregate approxXimation (SAX) representation [14]. First, we aggregate RSSI values by taking the average of each *aggregationWindow*. Then, we split the range of RSSI values into equiprobable bins. The number of bins is what we call *alphabetSize* as each bin will be assigned a symbol or an alphabet character. At this stage, we adapt the SAX to capture and embed an important characteristic of the RSSI signal which is the sampling pattern (Figure 1). For that, we append an extra bin (with an extra alphabet symbol) to represent any *aggregationWindow* for which we don't have

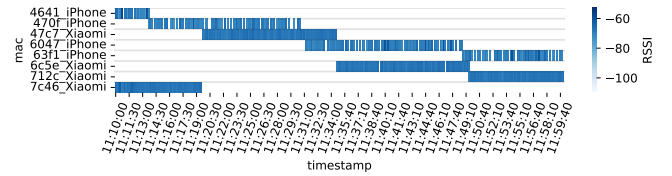


Fig. 1: Bluetooth signals from the same device can have a unique sampling pattern that differentiates it from other devices. Xiaomi RSSI time-series are denser compared to the ones of the iPhone. This property can be used as a feature to link RSSI time-series to the same device.

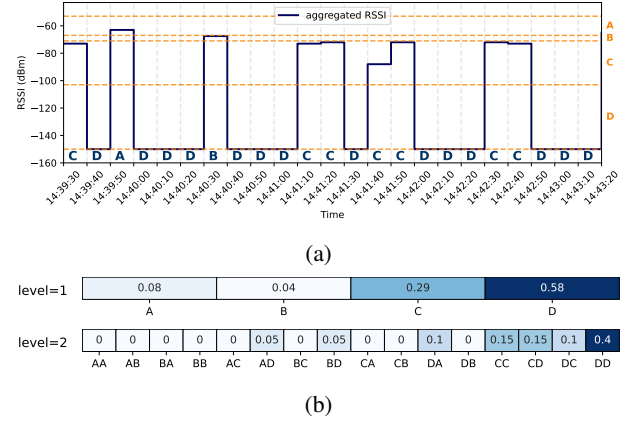


Fig. 2: (a) Transforming an RSSI time-series into a sequence of symbols "CDADD...CDDDA" using SAX (*alphabetSize* = 3). (b) SAX vectors with different levels *level* = 1 and *level* = 2.

RSSI values. The empty *aggregationWindows* are attributed a valid, yet a very small, RSSI value (e.g. -150 dBm) to differentiate it from the measurable values, which are typically much higher than -150 dBm. By concatenating the symbols of successive *aggregationWindows*, we end up with a sequence of symbols as a SAX-representation of the time-series \mathcal{T}_M . The sequence is then split into words of size *wordSize*. The process of transforming the time-series into a SAX sequence is illustrated in Figure (2a).

Similar to forming IntelligentIcons [15] from the words of a SAX sequence, we form a 1-D vector that embeds the features of the time-series signal. We can control the level of resolution of the vectors by defining the *level* parameter. Figure (2b) shows 2 SAX vectors for the same time-series shown in Figure 2a at 2 different levels of resolution.

We can now identify time series with similar attributes by performing a pairwise comparison of their corresponding SAX-vectors. This can easily be done using cosine-similarity (Equation 2).

$$\cos(\mathcal{V}^{M1}, \mathcal{V}^{M2}) = \frac{\sum_{i=1}^n \mathcal{V}_i^{M1} \mathcal{V}_i^{M2}}{\sqrt{\sum_{i=1}^n (\mathcal{V}_i^{M1})^2} \sqrt{\sum_{i=1}^n (\mathcal{V}_i^{M2})^2}} \quad (2)$$

B. Matching time-series from the same device

1) *Selecting the candidates:* When a device uses MAC address randomization, it generates a new random address

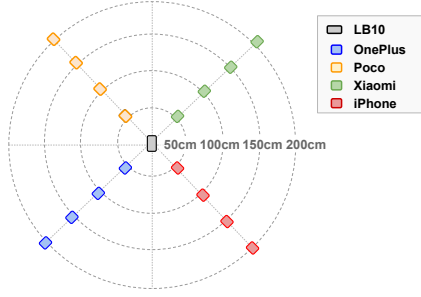


Fig. 3: Experimental design.

M_{new} to replace the previous one M_{old} . We are interested in any new MAC address that appears closely following the disappearance of M_{old} .

We define a delay δ_{delay} to be the tolerance delay between the last timestamp of $\mathcal{T}^{M_{old}}$ and the first timestamp of any new time-series. Instead of assigning a fixed value for δ_{delay} , we adapt it to the characteristics of \mathcal{T}^M [11], as each device may have different delays for generating new MAC address.

Let Δ_t^M be a set of time differences between successive timestamps in \mathcal{T}^M .

$$\Delta_t^M := \{t_{i+1}^M - t_i^M : i = 1, 2, 3, \dots\} \quad (3)$$

Then, δ_{delay} is defined as follows.

$$\delta_{delay} := 2 \cdot \max(\Delta_t^{M_{old}}) \quad (4)$$

We define the set of candidates \mathcal{S}_{cand} as follows.

$$\mathcal{S}_{cand} := \{m_{cand} | \min(t^{m_{cand}}) - \max(t^{M_{old}}) \leq \delta_{delay}\} \quad (5)$$

From that, we compute the SAX vectors for every element in \mathcal{S}_{cand} as well as for M_{old} .

2) *Assigning the best candidate:* After forming the set of candidates \mathcal{S}_{cand} , we compute a pairwise cosine similarity between the SAX vector of M_{old} and every candidate in \mathcal{S}_{cand}

$$SIM(M_{old}) = \{\cos(\mathcal{V}^{M_{old}}, \mathcal{V}^{m_{cand}}); \forall m_{cand} \in \mathcal{S}_{cand}\} \quad (6)$$

We can repeat the same process described in Equations (3) to (6) for every disappearing MAC address in \mathcal{D} . Then, assign the best candidate to each M_{old} by solving a linear assignment problem [11] that maximizes the sum of similarities.

III. EXPERIMENTAL EVALUATION

A. Experimental setup

To evaluate the performances of McMatcher for matching and linking time-series with different random MAC addresses transmitted by the same device, we propose the experimental setup shown in Figure 3.

The experiment consists of putting 4 different smartphones with MAC address randomization on a table at different distances with respect to the measuring device LB10 (sniffer). We used 3 Android smartphones (Poco X4, OnePlus 6, Xiaomi Mi MIX2) and an iPhone 13 pro. The smartphones were put at equidistance from LB10, and the same experiment was

repeated 4 times to cover the 4 distances (50cm, 100cm, 150cm, and 200cm). Each experiment lasted about 1 hour. The experiments were performed in an indoor office environment.

B. Labelling smartphone data

In order to be able to evaluate the performance of our methodology, we need to label the BLE signals from each smartphone to collect the ground truth. To do so, we install a beacon simulator mobile application [16, 17] on the smartphones. Once installed, we manually modify the advertising payload to include a unique pattern for each smartphone. Once the setup is ready, we start the beacon simulator to start generating the BLE signals.

C. The sniffer

We use an ESP32-based device, labeled LB10 in Figure 3, to collect the data. We instrumented LB10's firmware to continuously perform active BLE scans (realtime sampling rate). The received BLE beacon broadcast by the smartphones includes the randomized MAC address of the transmitting smartphone, the signal strength (RSSI), and the advertisement payload. The results of the scans are timestamped and stored locally as CSV files inside an SD card storage on LB10.

D. Combining multiple experiments to create a rich dataset

We combine the 4 experiments in order to emulate the presence of the 4 smartphones at different distances simultaneously. We do that by realigning the data to start all at the same time. We obtain 16 smartphones around the sniffer (4 smartphones x 4 distances), as shown in Figure 3. In addition to the 4 smartphone devices, there were other Bluetooth devices incidentally present in the background that can also use MAC randomization, and generate more MAC addresses that can be detected by the sniffer. We keep the data from those extra devices when evaluating McMatcher.

E. Evaluation metrics

To evaluate the performance of McMatcher in matching randomized MAC addresses that are generated by the same smartphone, we define the following metrics.

- True Positive (TP): when the assigned MAC address by McMatcher is the actual MAC address generated by the same smartphone after the MAC address M_{old} .
- True Negative (TN): when there is *no new* MAC address that comes after the MAC address M_{old} and McMatcher *does not assign* a new MAC address.
- False Positive (FP): when there is *no new* MAC address that comes after the MAC address M_{old} but McMatcher assigns a MAC address from another device.
- False Negative (FN): when *there is* a new MAC address that comes after the MAC address M_{old} but McMatcher *does not assign* any MAC address, *or assigns* a MAC address from another device.

From which we can define the following.

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP} \quad (7)$$

Distance (cm)	Duration	OnePlus	Poco	Xiaomi	iPhone
50	01:00:50	1344	1383	933	730
100	01:25:00	3057	2663	2533	797
150	01:00:00	1376	1872	1327	599
200	01:00:00	1381	1239	1762	769

TABLE I: Duration of each experiment and the number of RSSI samples collected by each device.

Device Distance	OnePlus	Poco	Xiaomi	iPhone	total(distance)
50	6	6	5	5	22
100	8	8	6	6	28
150	6	5	5	5	21
200	6	6	5	4	21
total/device	26	25	21	20	92

TABLE II: MAC addresses generated by each smartphone.

$$precision = \frac{TP}{TP + FP} \quad recall = \frac{TP}{TP + FN} \quad (8)$$

$$F1\text{-score} = \frac{2 * precision * recall}{precision + recall} \quad (9)$$

IV. RESULTS AND DISCUSSION

A. The dataset characteristics

Table I shows the number of RSSI samples and the total duration of the experiments. Each experiment lasted at least 1 hour, and each smartphone generated more than 1000 RSSI samples in almost every experiment, except for the iPhone, which generated the least number of samples.

By combining the 4 experiments into a single dataset, we obtain a dataset containing **332** unique MAC addresses, out of which, **92** MAC addresses belong to the 4 smartphones that we used in our setup, which represents 30% of MAC addresses in the dataset, the other MAC addresses are from the background devices incidentally present during the data collection. Table II shows the number of unique random MAC addresses generated by each smartphone at different distances from the sniffer LB10. Each smartphone generated between 4 to 8 random MAC addresses, resulting in more than 20 unique random MAC addresses at each distance. We also show in Figure 4a, the lifespan of the MAC addresses. The smartphone MAC addresses have a similar lifespan to other MAC addresses in the dataset, with durations ranging between 3 to 20 minutes.

Figure 4c shows the amount of time (delay) it takes for the new random MAC address, generated by the smartphones, to appear in the measurements after the old MAC address has disappeared. The OnePlus and the Poco smartphones have relatively shorter delays (less than 10s for 60% of the cases) compared to the Xiaomi (mostly between 10s-20s delay) and the iPhone, which shows even longer delays (40s-50s). In terms of the time difference between consecutive RSSI samples from each smartphone (sample rate), the 3 Android smartphones have more frequent samples compared to the iPhone samples, as shown in Figure 4b. Overall, the larger the time difference between RSSI values from a device, the longer

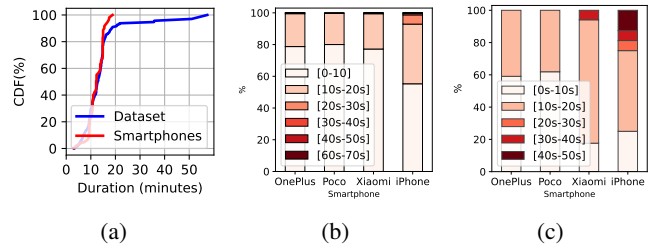


Fig. 4: (a) MAC addresses lifespan. (b) Mac address randomization delay (in seconds). (c) Time difference (in seconds) between RSSI samples from each smartphone.

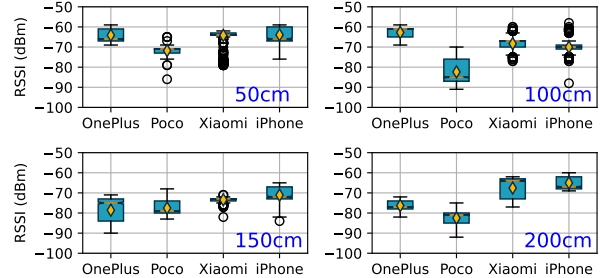


Fig. 5: RSSI distribution.

the delay for the new random MAC address to be discovered by the sniffer, as previously suggested in [11].

We also show in Figure 5, the distribution of the RSSI measurements from each smartphone for the 4 experiments. Across different distances tested, or within each experiment, we see that some smartphones can have similar RSSI distributions. This suggests that relying solely on aggregated RSSI statistics (e.g. average RSSI) to compare signals for matching random MAC addresses might not be enough.

B. McMatcher matching

We apply the McMatcher methodology described in Section II to our dataset with the following (SAX [14]) parameters: $\{aggregationWindow = 10s, alphabetSize = 5, wordSize = 6, level = 2\}$. We obtain a 93% accuracy with a high precision (99%), a low recall (93%), and an F1-score of 96%. The matching errors (6), shown in Table III, are mostly FN errors (5), and most of these errors are for the 150cm distance. We can also see that the errors are among smartphones with similar RSSI distributions, as shown in Figure 5. This suggests the model is not capable of distinguishing between those smartphones with the current values for the parameters.

We need to tune the parameters of the model by making it more sensitive to the differences in the RSSI signals from the different smartphones in order to achieve better results. For

mac	next_mac	predicted	Error
57f9_iPhone1.5	5013_iPhone1.5	NaN	FN
5013_iPhone1.5	NaN	528c_UNKN1.0	FP
7240_iPhone2.0	5316_iPhone2.0	6202_OnePlus0.5	FN
66c9_OnePlus0.5	6202_OnePlus0.5	5316_iPhone2.0	FN
7b0a_OnePlus1.5	52ab_OnePlus1.5	6f95_Poco1.5	FN
5007_Poco1.5	6f95_Poco1.5	52ab_OnePlus1.5	FN

TABLE III: Mismatched results. The errors are mainly FN.

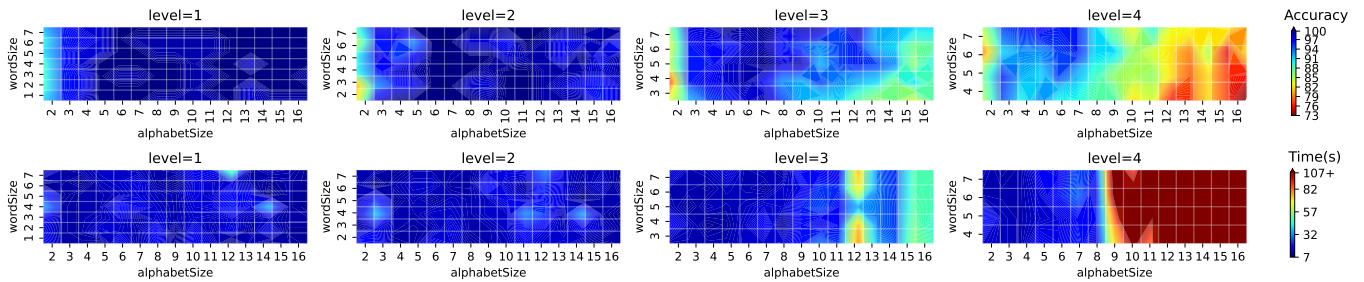


Fig. 6: Accuracy and computation time for different combinations of McMatcher parameters.

that evaluate the same set of metrics defined in Equations (7) to (9), in addition to the computation time, against different combinations of values for the parameters $alphabetSize$, $wordSize$, and $level$. The results are shown in Figure 6. The evaluation was done on a Dell laptop with an 11th Gen Intel® Core™ i7-1185G7 @ 3.00GHz × 8.

Impact of the $level$ parameter. By increasing $level$, we increase the level of details of the SAX-vectors, as we already discussed in Section II and as shown in Figure 2b. This means more intrinsic patterns characterizing the signal are embedded in the vectors. So, a higher $level$ value will make the vectors more unique, which will result in lowering the similarity, which in turn, will have a negative impact on the accuracy of the matching. As we can see in Figure 6, the accuracy of the model degrades as we increase the $level$ value. In terms of computation speed, higher $level$ values, especially higher than 3, increase the computation time when combined with larger $alphabetSize$ values (more than 8). This is because when a high $level$ value is combined with a high $alphabetSize$ will result in longer vectors, which makes the cosine similarity in Equation 2 more time-consuming.

Impact of the $alphabetSize$ parameter. From Figure 6, the accuracy increases by increasing the $alphabetSize$. Overall, the accuracy is highest with $alphabetSize$ 6 and 7. The accuracy then decreases gradually by increasing $alphabetSize$. This effect is more noticeable with higher values of $level$. From our discussion in Section II, $alphabetSize$ controls the number of bins of the RSSI range. Smaller values produce fewer bins with larger bin ranges, which results in vectors with low resolution. The model will not be able to capture the variability of the RSSI values which makes it less tolerant to the changes in the RSSI signal. In contrast, a very high $alphabetSize$ value will make the bins smaller, and higher resolution vectors. This means capturing more variability of the RSSI values and making the vectors more unique, which will result in a similar impact as the $level$ parameter that we discussed earlier. For computation speed, we get a longer computation time at higher $alphabetSize$ values due to the same reasons we discussed for the impact of $level$ parameter.

Impact of the $wordSize$ parameter. This parameter has the least impact compared to $level$ and $alphabetSize$ parameters. However, it's tightly related to the $level$ parameter when forming the vectors. For smaller $level$ (less than 3), the $wordSize$ has almost no impact. For $level$ values more

accuracy	precision	recall	$F1$ -score
1.0	1.0	1.0	1.0

TABLE IV: Improved McMatcher results with parameters $alphabetSize = 7$, $wordSize = 6$, $level = 2$.

Sampling rate (s)	Realtime	10	20	30	40	50
Accuracy	1.0	0.98	0.91	0.92	0.89	0.88

TABLE V: Impact of sniffer sampling rate on the accuracy.

than 2, the accuracy increases as the $wordSize$ increases. This can be related to how the SAX-vectors are formed depending on $wordSize$. We need longer words in order to be able to extract more sub-sequences of length of $level$. Similarly, for computation time, $wordSize$ has a negligible impact.

As shown in Figure 6, McMatcher can achieve **100%** accuracy for many combinations of the parameters. From our previous discussion on the impact of each parameter, we can choose $alphabetSize = 7$, $wordSize = 6$, $level = 2$. The accuracy, precision, recall, and the F1-score of the matching with these values are shown in Table IV. McMatcher achieves this level of accuracy using the RSSI only, and without any prior model training, as opposed to the work presented in [13], which achieves 99.5% accuracy on the same dataset, but does require training a machine learning model.

Impact of the sniffer's sampling rate. After improving the model's results. We now evaluate the impact of the sniffer's sampling rate on the accuracy of matching the MAC addresses. We keep the optimal parameters for McMatcher obtained in the previous analysis, and we reduce the sampling rate of the sniffer. The results are shown in Table V. Taking 1 sample every 10s achieves 98% accuracy, slightly lower than the best accuracy (100%) obtained with the realtime, continuous scanning. Lowering the sampling rate results in lowering the accuracy. Realtime scanning is recommended for optimal performance. However, in devices for which power consumption and battery life are a concern, a good trade-off between the sampling rate and accuracy is needed.

C. Realtime matching performances

As opposed to the methodology presented in [13], our methodology does not require training a machine learning model and only uses the cosine similarity for matching randomized BLE MAC addresses. McMatcher can be applied in realtime by building vectors and selecting the candidates

as newer MAC addresses start appearing in the data. The McMatcher will then work on a subset of MAC addresses which will take even less time to compute. By making a sequential processing (realtime) on our dataset, the model manages to achieve 100% accuracy by solving smaller, sub-matching problems in just **230 milliseconds** on average, with around 18 MAC addresses in each sub-problem. This makes McMatcher easier to deploy on power-restricted devices such as IoT devices, embedded devices, and wearables.

V. CONCLUSION

In this paper, we presented McMatcher, a novel methodology that uses a symbolic representation of RSSI time-series (SAX) to build characterizing vectors for the random BLE MAC addresses data. The vectors can then be compared using cosine similarity. McMatcher achieves 100% accuracy on a set of different brand smartphones that use MAC address randomization by successfully matching 92 random MAC addresses to 16 smartphones in a dataset containing 332 MAC addresses in total. McMatcher achieves this accuracy using only the RSSI data without any model training, and can work on any transmitter data regardless of the BLE protocol version. In addition to that, it offers a matching quality metric that can be used to express the matching confidence level. This makes our work a novel way for matching random BLE MAC addresses. Future works will emphasize on evaluating McMatcher in dynamic scenarios, in which the RSSI signal can have high temporal heterogeneity. In such scenarios, we apply McMatcher on the the head and the tail of the new and old MAC address time-series, respectively, to have a more robust matching to cope with the dynamic nature of the environment. Preliminary results already show high accuracy even in dynamic environments (100% accuracy in two bus trips with two transmitters as target devices).

REFERENCES

- [1] K. Townsend, C. Cufí, R. Davidson *et al.*, *Getting started with Bluetooth low energy: tools and techniques for low-power networking*. " O'Reilly Media, Inc.", 2014.
- [2] "2023 Market Update | Bluetooth® Technology Website," Sep. 2023, [Online; accessed 10. Sep. 2023]. [Online]. Available: <https://www.bluetooth.com/2023-market-update>
- [3] K. Fawaz, K.-H. Kim, and K. G. Shin, "Protecting privacy of {BLE} device users," in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 1205–1221.
- [4] J. K. Becker, D. Li, and D. Starobinski, "Tracking anonymized bluetooth devices." *Proc. Priv. Enhancing Technol.*, vol. 2019, no. 3, pp. 50–65, 2019.
- [5] P. Locatelli, M. Perri, D. M. Jimenez Gutierrez, A. Lacava, and F. Cuomo, "Device discovery and tracing in the Bluetooth Low Energy domain," *Comput. Commun.*, vol. 202, pp. 42–56, Mar. 2023.
- [6] G. Kalantar, A. Mohammadi, and S. N. Sadrieh, "Analyzing the effect of bluetooth low energy (ble) with randomized mac addresses in iot applications," in *2018*

- IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (Smart-Data)*, 2018, pp. 27–34.
- [7] B. Sowmiya, V. S. Abhijith, S. Sudersan, R. Sakthi Jaya Sundar, M. Thangavel, and P. Varalakshmi, "A Survey on Security and Privacy Issues in Contact Tracing Application of Covid-19," *SN Comput. Sci.*, vol. 2, no. 3, pp. 1–11, May 2021.
- [8] M. Cunche, A. Boutet, C. Castelluccia, C. Lauradoux, and V. Roca, "On using bluetooth-low-energy for contact tracing," Ph.D. dissertation, Inria Grenoble Rhône-Alpes; INSA de Lyon, 2020.
- [9] J. K. Becker, D. Li, and D. Starobinski, "Tracking Anonymized Bluetooth Devices," *Proceedings on Privacy Enhancing Technologies*, vol. 2019, no. 3, pp. 50–65, Jul. 2019.
- [10] G. Celosia and M. Cunche, "Saving private addresses: an analysis of privacy issues in the bluetooth-low-energy advertising mechanism," in *MobiQuitous '19: Proceedings of the 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*. New York, NY, USA: Association for Computing Machinery, Nov. 2019, pp. 444–453.
- [11] L. Jouans, A. C. Viana, N. Achir, and A. Fladenmuller, "Associating the randomized bluetooth mac addresses of a device," in *2021 IEEE 18th Annual Consumer Communications & Networking Conference (CCNC)*, 2021, pp. 1–6.
- [12] S. Akiyama, R. Morimoto, and Y. Taniguchi, "A study on device identification from ble advertising packets with randomized mac addresses," in *2021 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia)*, 2021, pp. 1–4.
- [13] G. Gagnon, S. Gambs, and M. Cunche, "Rssi-based fingerprinting of bluetooth low energy devices," in *International Conference on Security and Cryptography (SECRYPT 2023)*, 2023.
- [14] J. Lin, E. Keogh, L. Wei, and S. Lonardi, "Experiencing sax: a novel symbolic representation of time series," *Data Mining and knowledge discovery*, vol. 15, pp. 107–144, 2007.
- [15] E. Keogh, L. Wei, X. Xi, S. Lonardi, J. Shieh, and S. Sirowy, "Intelligent icons: Integrating lite-weight data mining and visualization into gui operating systems," in *Sixth International Conference on Data Mining (ICDM'06)*, 2006, pp. 912–916.
- [16] "Beacon simulator - apps on google play," Sep. 2023, [Online; accessed 4. Sep. 2023]. [Online]. Available: <https://play.google.com/store/apps/details?id=net.alea.beaconsimulator>
- [17] "Beacon Simulator," Sep. 2023, [Online; accessed 18. Sep. 2023]. [Online]. Available: <https://apps.apple.com/lb/app/beacon-simulator/id1380778696>