A System for Privacy-Preserving Analysis of Vehicle Movements

Gianluca Lax(✉), Francesco Buccafurri, Serena Nicolazzo, Antonino Nocera, and Filippo Ermidio

DIIES, University Mediterranea of Reggio Calabria, Via Graziella, Località Feo di Vito, 89122 Reggio Calabria, Italy
lax@unirc.it

Abstract. In this paper, we deal with the problem of acquiring statistics on the movements of vehicles in a given environment yet preserving the identity of drivers involved. To do this, we have designed a system based on an embedded board, namely Beaglebone Black, equipped with a Logitech C920 webcam with H.256 hardware encoder. The system uses JavaANPR to acquire snapshots of cars and recognize license plates. Acquired plate numbers are anonymized by the use of hash functions to obtain plate digests, and the use of a salt prevents plate number discovery from its digest (by dictionary or brute force attacks). A recovery algorithm is also run to correct possible errors in plate number recognition. Finally, these anonymized data are used to extract several statistics, such as the time of permanence of a vehicle in the environment.

Keywords: Privacy · Vehicle movements · Beaglebone · JavaANPR

1 Introduction

In the smart city's evolution, embedded systems have played a smaller but no less important role [1,2]. Daily life is full of these systems, we do not see and/or notice them but they exist and they are growing in number: ATMs, washing machines, navigators, credit cards, temperature sensors and so on. Data automatically collected by embedded devices (e.g., sensors) has a great value: typically, such data are processed and transformed into information (knowledge) thanks to which we can make decisions that may or not require human participation.

In this paper, we present a system able to collect data of vehicle movements that can be used for analysis purposes. The system is designed to overcome possible privacy concerns arising from collecting and processing of data linked to one individual (i.e., vehicle driver) by the license plate of the vehicle, a problem very relevant in the literature [3–8]. In particular, we created a license plate recognition system to track vehicles entering or leaving a particular place. Plates are not stored in plaintext: an approach based on salt and hash is adopted to transform plain plate into an apparently random string. However, the approach is such that the same plate will be transformed into the same string each time

A. Longo et al. (Eds.): IISSC 2017/CN4IoT 2017, LNICST 189, pp. 23–28, 2018.
https://doi.org/10.1007/978-3-319-67636-4_3

Fig. 1. An example of the system utilization.

the vehicle is tracked. This allows us to enable statistical analysis on stored data yet maintaining anonymity of drivers and vehicles.

The rest of the paper is organized as follows: in the next section, we describe the system architecture, the hardware components and the executed protocols; in Sect. 3, we discuss advantages and limitations of our proposal and draw our conclusions.

2 System Architecture and Implementation

In this section, we describe the architecture of our system and the algorithms used to solve the problem.

In Fig. 1, we sketch a simple example of the use of our system. We consider a closed environment, a parking in the figure, where cars enter and exit periodically and we need to know some statistics about users' habit, for example, the minimum, maximum and average time of permanence of a vehicle in this area. An additional constraint is that the solution has not to reveal any information about any specific vehicle, for privacy reasons. Consequently, solutions based on RFID or similar technologies to recognize a vehicle cannot be adopted.

In the figure, a device placed at the enter/exit of the parking is also shown (it is represented as a simple camera). This device is the system proposed in this paper to solve the problem. Our system is built on the BeagleBone platform [9], a single-board computer equipped with open-source hardware. We used the Beaglebone Black version, a low-cost high-performance ARM device with full support for embedded Linux. It is a perfect device for interfacing to low-level hardware, while providing high-level interface in the form of GUIs and network services. With a price of about 50$ and a clock speed of 1GHz, it is a cheap solution capable of significant data processing tasks. The BeagleBone Black used in

Fig. 2. The BeagleBone and the Logitech C920 webcam.

our proposal is equipped with a high resolution Logitech C920 webcam (Fig. 2), which contains a H.256 hardware encoder to take the workload away from BeagleBone's processor. The Video4Linux2 (typically called V4L2), a framework tightly integrated with the Linux kernel, provides drivers necessary for the webcam. Our BeagleBone runs a software implementing the data processing logic that allows us to obtain privacy preserving logs of vehicle entry and exit. In particular, each time a vehicle enters or leaves the environment, Algorithm 1 is executed.

In the initialization phase, the system randomly generates a 256-bit string, named *salt*, and allocates a persistent memory area, named *log*, in which statistics on vehicles movements are stored (typically, this is a file). When a vehicle enters or leaves the environment, a picture is captured by the webcam and, then, a plate number recognition procedure is run (Line 1). This procedure uses JavaANPR [10], an automatic number plate recognition software, which implements algorithmic and mathematical principles from the field of artificial intelligence, machine vision and neural networks. In case the vehicle is entering the environment, the system computes the hash of the string obtained by concatenating the salt and the binary representation of the plate number (Line 3).

Concerning this operation, we observe that several hash functions can be used in this task: for our purpose, we need that it is not possible to find the salt or p from the knowledge of more hashes. In our implementation, we opted for the SHA-1 algorithm, which is a widely used hash function producing a 160-bit hash value [11]. Indeed, although SHA-1 has been found to suffer from some vulnerabilities that would discourage its use as a cryptographic hash function, in our application such vulnerabilities are not critic. Moreover, its efficiency and effectiveness to verify data integrity, make it a good solution for our necessity.

Then, the algorithm proceeds by storing into the log a tuple containing the type of access of the vehicle (enter of exit), the timestamp of this access, and the result of the hash computation, named p^* (Line 4). This tuple is one of the records that can be elaborated to extract statistics about vehicle accesses. Observe that, no reference to the actual plate number is stored, but only the hash of this number. Moreover, the use of a *salt* [12] in the hash computation protects against dictionary attacks versus a list of password hashes and against pre-computed rainbow table attacks, aiming at guessing the plate number.

Consider now the case in which a vehicle is leaving the environment. In the optimistic case in which the plate number recognition task can be performed

Algorithm 1. *Log creation*

Constant *salt*: a 256-bit string
Constant log: a persistent memory area
Input *entry*: a boolean (true in case of vehicle entry, false otherwise)
Variable T: the current timestamp
Variable p: a number plate
Variable p^*: a 64-bit string
Variable P: the set of number plates differing at most 1 digit from p

```
 1: p = plate number recognition
 2: if entry = true then
 3:     p* = H(salt || p)
 4:     append ⟨entry,T,p*⟩ into LOG
 5: else
 6:     create P from p
 7:     for all p ∈ P do
 8:         p* = H(salt || p)
 9:         if p* is in LOG then
10:             append ⟨entry,T,p*⟩ into LOG
11:             return
12:         end if
13:     end for
14:     append ⟨entry,T,-1⟩ into LOG
15: end if
```

with no error (i.e., the recognized number p coincides with the actual number of the plate), then it would be sufficient to repeat the operation above and storing into the log the information *exit* instead of *entry*. However, it is possible that some errors occur in plate number recognition. Consequently, we included in the algorithm a procedure to mitigate the consequences derived from this error. Specifically, the first operation done is to compute the set P (Line 6), composed of all plate numbers differing from p at most of 1 digit.

Now, for each element p of the set P, the hash p^* is computed as done above (Line 8) and a search for this value in the logs related to previous vehicle entries is carried out (Line 9). If a match is found, then the tuple containing the information about the exit of the vehicle, the current timestamp, and the result of the hash computation p^* is stored and the algorithm ends (Lines 10 and 11). In words, this operation allows the system to identify the right matching between the vehicle entry and exit (recall that the actual plate number is never recorded – thus, this task is not trivial), even when at most 1 digit of the plate number is wrongly recognized.

Finally, in case no matching is found, the information that an unidentified vehicle (this is coded by using -1 as plate number) is leaving the environment, is stored (Line 14).

At the end of the monitoring period, the log will contain a list of accesses of vehicles to the environment, together with the access timestamp and an *anonymous* reference to the number plate.

This list can be used to infer several statistics about the vehicle accesses, such as the minimum or maximum permanence period of a vehicle in the environment (how to calculate such statistics from this list is a trivial exercise and is not discussed here).

3 Discussion and Conclusion

In this section, we briefly discuss some aspects related to our proposal that have been neglected for space limitations. We designed and implemented a BeagleBone-based system to monitor vehicle entry to and exit from an environment with the purpose of maintaining a privacy-preserving log of these accesses. By this log, it is possible to calculate statistics about the period of presence of a vehicle in this environment (for example, the minimum or maximum permanence). As in a pervasive environment there is no knowledge about vehicles entering the environment, the statistics described above cannot be obtained through a solution different from the one proposed in this paper, if the constraint of not storing any number plate has to be guaranteed. Indeed, the only (usable) identifier is the plate number because the pervasiveness of the scenario inhibits the adoption of a solution based on the use of RFID technology or similar (it is unrealistic to provide any vehicle with an RFID tag).

In our proposal, no plate number is stored in clear, in such a way that any attack on the storage device (by a malware, for example) cannot infer any information about the plate number. Moreover, the use of the SHA-1 algorithm and the salt prevents an attacker to guess the stored plate number. It is worth noting that the well-known (collision and pre-image) vulnerabilities of SHA-1 in our system do not give an attacker any advantage. Indeed, an attack should try a brute force attack on a 256-bit string to guess the salt, and this is currently considered unfeasible.

Another important aspect is that we cannot assume the operation of plate number recognition is carried out with no error. In our proposal, the knowledge about the plate number (or better, its hash) of the vehicles already present in the environment is used to guess the correct plate number in case of recognition error. In our proposal, we limit to 1 the number of errors recoverable (i.e., we guess a plate number only when the recognized plate number differs from the actual one of at most 1 digit). Observe that, this requires to generate and test (recall Lines 6–13 of Algorithm 1) a very limited number of possible plate numbers: this number depends on the country and, for many EU countries, it is about 115. As the computation of SHA-1 is an efficient task [13], the running time of the algorithm is also very limited (much less than 1 s). Concerning this aspect, a preliminary experimental evaluation (which is only summarized here for space constraint) reported that on average 73% plate numbers are recognized with no error, and that in 98% cases the recognition task produced at most 1 digit error (we recall that our algorithm is designed in such a way to tolerate 1-digit errors). This allows us to state that the statistics produced starting from the data collected by our system can be considered quite accurate, as in 98% cases, the matching between entering and leaving cars is correctly carried out.

A possible improvement of this study, which is left as a future work, is a large-scale testing of the system in more *complex* environments, where the presence of highly dynamic accesses and worse conditions (for example, of illumination) may result in a worsening of the plate number recognition task and, consequently, in a general detriment of the system performance. A second improvement regards the

relatively naive binary strategy of assigning label unknown to an exiting car not matching any entered car. Specifically, assigning probabilities for which car exits could be a significant improvement especially as these uncertainties could be reduced with multiple returns of the same vehicle. Finally, another improvement is related to the use of different and possibly more effective (than JavaANPR) techniques for plate number recognition, in order to further increase the overall performance of the system.

References

1. Filipponi, L., Vitaletti, A., Landi, G., Memeo, V., Laura, G., Pucci, P.: Smart city: An event driven architecture for monitoring public spaces with heterogeneous sensors. In: 2010 Fourth International Conference on Sensor Technologies and Applications (SENSORCOMM), pp. 281–286. IEEE (2010)
2. Merlino, G., Bruneo, D., Distefano, S., Longo, F., Puliafito, A., Al-Anbuky, A.: A smart city lighting case study on an openstack-powered infrastructure. Sensors **15**(7), 16314–16335 (2015)
3. Hoh, B., Iwuchukwu, T., Jacobson, Q., Work, D., Bayen, A.M., Herring, R., Herrera, J.C., Gruteser, M., Annavaram, M., Ban, J.: Enhancing privacy and accuracy in probe vehicle-based traffic monitoring via virtual trip lines. IEEE Trans. Mob. Comput. **11**(5), 849–864 (2012)
4. Li, H., Dán, G., Nahrstedt, K.: Portunes: privacy-preserving fast authentication for dynamic electric vehicle charging. In: 2014 IEEE International Conference on Smart Grid Communications (SmartGridComm), pp. 920–925. IEEE (2014)
5. Wu, Q., Domingo-Ferrer, J., González-Nicolá, Ú.: Balanced trustworthiness, safety, and privacy in vehicle-to-vehicle communications. IEEE Trans. Veh. Technol. **59**(2), 559–573 (2010)
6. Zhang, T., Delgrossi, L.: Vehicle Safety Communications: Protocols, Security, and Privacy, vol. 103. Wiley, Hoboken (2012)
7. Buccafurri, F., Lax, G., Nicolazzo, S., Nocera, A.: Comparing twitter and facebook user behavior: privacy and other aspects. Comput. Hum. Behav. **52**, 87–95 (2015)
8. Buccafurri, F., Lax, G., Nocera, A., Ursino, D.: Discovering missing me edges across social networks. Inf. Sci. **319**, 18–37 (2015)
9. BeagleBoard: Beagle Board Black Website (2016). http://beagleboard.org/BLACK
10. JavaANPR: Automatic Number Plate Recognition System (2016). http://javaanpr.sourceforge.net
11. Wikipedia: SHA-1 – Wikipedia, The Free Encyclopedia (2016). https://en.wikipedia.org/wiki/SHA-1
12. Wikipedia: Salt (cryptography) – Wikipedia, The Free Encyclopedia (2016). https://en.wikipedia.org/wiki/Salt_(cryptography)
13. Gosselin-Lavigne, M.A., Gonzalez, H., Stakhanova, N., Ghorbani, A.A.: A performance evaluation of hash functions for IP reputation lookup using bloom filters. In: 2015 10th International Conference on Availability, Reliability and Security (ARES), pp. 516–521. IEEE (2015)