# SafePaths Privacy Preserving WiFi Co-location with Prior Scanning

Camera Culture Group, MIT Media Lab
MIT Alliance on Distributed and Private Machine Learning
SafePaths Alliance

Version 0.1, May 14th, 2020

**Problem:**
Estimate whether two devices are close to each other (and the distance between them) while maintaining privacy using WiFi protocols.

**Motivation:**
Estimating location of a device with respect to a single of multiple WiFi access points from WiFi signals is a well studied topic. However, estimating co-location or proximity of two devices is a different problem. Generally, the (approximate) location is estimated using the presence of known WiFI SSID vectors of Access Points (APs) and a crowdsourced lookup table of WiFi access point MAC addresses and their GPS locations. For more precision, such estimation requires prior recording of signal strength with respect to WiFi APs or some geometric mapping in the known space using ground truth data.

We want to achieve co-location under two constraints (i) privacy (ii) no crowdsourced pre-recorded SSID/Mac addresses for signal strength or Mac address mapping to GPS locations.

**Strawman Algorithm and Privacy Challenges:**
A simple algorithm would store Wifi SSID/MAC address and RSSI vector on a central server, and compare the two in a centralized way.

Because of **privacy**, we cannot allow the two phones to reveal anything about themselves, other than the random numbers (rotating MAC addr), or some numbers like physical signal strength. For example, revealing proximity to a fixed AP MAC address with time stamp would reveal the time stamped GPS coordinates, and they would in turn would identify the person. We also do not want a centralized algorithm that will force even the healthy(or exposed) individuals to upload information from their phones about MAC addresses of the AP seen by each phone because that will reveal proximity of each healthy person's phone to the other.

Because of computational **complexity**, we cannot allow all phones to store this SSID-RSSI vector because m phones will create m^2 exchanges.

We do not expect immediate change to WiFi AP points protocols or changes to their firmware.

**Assumptions/Requirements**

AP mac address remains constant for a finite period (or for a long time)
Phone mac address change continuously, but phone app knows the self mac address
Phone can scan nearest several AP mac addresses and RSSI
Phone can use a different IP protocol while still scanning for nearest several AP mac addresses (we need the co-location calculation even if Phone is connected to a single AP)

**Techniques**

We consider two main techniques

1. Phone-driven: Smartphone app records (one or more) AP MAC addresses and compares with AP MAC addresses of other phones. No mac address of the smartphones is revealed. Infected phone uploads hash of the recorded AP addresses to the server, healthy phone downloads and compares. Optionally, an external server provides IP address specific tokens for additional privacy.
2. AP driven: AP captures MAC addresses of all phones in its proximity in a given time slot and uploads to the server as a proximate group of phones with timeslot. MAC addresses of phones are continuously random. Each phone also stores its own mac address at each instant. Infected phone uploads a series of timestamped MAC self-addresses. Server finds the AP that had seen this infected phone. The server identifies proximate group of phones that matches the MAC address of the inflected phone and also computes distances based on RSSI. Server marks all time stamped MAC addresses of phones in the given exposed proximate groups as exposed. Healthy phone downloads an all time stamped phone MAC address and checks if the self-MAC address is part of this downloaded set.

**Option 1: Phone Driven upload of AP MAC addresses**

Description:

P - Healthy phone, Q - Infected phone
Space of MAC address is 12 Hexadecimal codes, which is 16^12 = 2^48
Example MAC address is 73:47:A1:27:DC:26
Standard Databases will be used for efficient query lookup over the hashed MAC addresses.
Note on the 802.11 beacon protocol - SSID is of variable length and can be upto 32 bytes. We require 128 bits in total for encrypting text while a MAC address only gives 48 bits. Leaving need for 80 bits more for sufficiently large entropy. These 80 bits can be

obtained by using the SSID which is expected to have 10 byte length at least. However the bits used for SSID are not truly random as they only contain readable characters, furthermore a SSID string can be assigned probability of occurrence based on the statistics of the words used in natural language.

The algorithm has the following steps.
(i) **Log**: Phones keep track of AP MAC address every one minute.
(ii) **Upload hash**: Infected phone uploads a hash of (MAC address, time stamp) and RSSI to the server and the server partitions the database for efficient lookup
(iii) **Bucketized query of hash**: Healthy phone downloads a small partition (bucket) of hash(MAC address, time stamp) and to see if there is a match with one or more AP MAC addresses. We use the first 20 bits for partition.
(iv) **Co-location**: Healthy phone computes the distance from the infected individuals based on RSSI from multiple AP sources within the same time bin .

A. Steps
    a. User Q and P record every MAC address x_i and SSID ssid_i of access point i at timestamp ts (rounded off to a minute) and store it in the following format:
        i. C = {[hash(x_1||ssid_1||ts), rssi], …}
    b. Q, once diagnosed positive, uploads their C_Q to the server S.
    c. S receives C_Q and appends it to its existing C_S and updates its database of bucket sets B_S with new entries coming from C_Q. This B_S database will be used for efficient constant time lookup.
    d. P uses their own locally stored set C_P to construct a **bucket** set B_P where B_P = {f([hash(ts||x||ssid)]), …} here, f(x) = first 20 bits of x.
    e. P sends B_P to the server S.
    f. S performs lookup in their database B_S for all b_P in B_P and returns set of elements in C_S corresponding to B_P ∩ B_S.
    g. P performs a full match with C_i's to find the right matches.
    h. P then feeds C_i ∩ C_P for all different time bins to *function **Localize**({rssi_i, ...}, {rssi_j, ...})* which calculates approximate distance between the two individuals based on rssi readings from multiple sources.

B. Benefit
    a. Health phone never uploads the seen MAC addresses and hence provides great privacy
    b. Only the bucket set B_P is uploaded by the P and obtaining C_P from B_P will have extremely low probability of success.

C. Challenges
    a. Infected person has to upload Hashed version of AP MAC addresses and this is susceptible to brute force search attack.
    b. Because a global crowdsourced mapping between MAC addr and GPS is available[1], If the server is able to bruteforce the AP mac addresses, it could obtain GPS coordinates of the infected person if it is available online.

c. However using 32 bit time stamp the search attack risk is reduced but not eliminated.

**Option 1B: External Server to improve entropy**

A. Use an external server to provide a unique random 128 bit secure key based on IP address to every P and Q at every minute. Rest of the operations are the same as Option 1.
    a. Q, hashes its AP mac address with 128 bit key and uploads
    b. P downloads this list and compares the hash by decrypting with their 128 bit keys received from the server
B. Benefit
    a. Removes the search attack that exposes GPS of AP based on AP's MAC address
    b. Attackers cannot fake the IP address to get the keys because the packets won't be delivered to them.
    c. This is a great solution for workplaces, universities as users will connect to the free enterprise wifi that has a single public IP address.
C. Challenges
    a. Requires the users to be constantly connected to internet to get 128 secure key every minute
    b. Requires access to the internet for both users via the same AP. If users are walking down a street together, and accessing the internet using Mobile data, the server will not be able to deliver the right keys as the IP address will be inconsistent.

**Option 2: AP driven upload of a group of proximate phones in Enterprise setting**

Description: The AP driven approach provides privacy to all users by realizing that in the enterprise WiFI settings, the AP keeps track of the MAC addresses of all devices they see. Since MAC addresses of the phones are randomized, the algorithm keeps track of 'proximity group', i.e. MAC addresses of all then phones seen by a single AP in a one minute time slot.

A. Steps:
    a. Every AP captures MAC addresses M_i of every phone i with timestamp t_i and rssi_i. The devices can rotate their MAC address for ensuring privacy.
    b. The AP sends this data to the log server which stores this data in a persistent storage.
    c. Each phone also stores its own randomized mac address at each instant.
    d. When a phone user gets identified as infected, the phone uploads the series of MAC addresses selectively. So the MAC address which the user advertised at company X will not be uploaded to the MAC address associated with server Y.

    e. On receiving a list of MAC addresses from a client, the server computes proximity group P which contains the list of other MAC addresses which should have been in a close vicinity of the infected person's MAC address. The server releases the proximity group P.

B. Benefits:
    a. Privacy of the infected individual is preserved as only the trusted AP control server knows the MAC address of the infected person. For the healthy individual only the MAC address is released in the public and under the assumption that this MAC address is also rotated and randomized by every user, the privacy leakage is minimal.

C. Challenges:
    a. The access point control server needs to be trusted in this case, making it a centralized algorithm. However, a majority of the publicly used access point control servers already log this data and the only additional information added in this protocol is the information about diagnosis or risk score linked with a list of rotating MAC addresses.

**Option 3: Wifi Broadcast (same as BLE)**

Description: This option uses the idea that a phone can also act as a WiFi broadcast channel temporarily and broadcast packets to its nearby devices. For receiving it acts as a scanner in normal mode. The privacy and information exchange steps here are the same as Bluetooth based contact tracing. We use random ids for transferring however, to make the method more efficient we can trivially substitute it with daily keys.

A. Steps:
    a. Every device broadcasts a random id and timestamp as part of their WiFi broadcast, for an interval of x ms.
    b. All devices in nearby vicinity receive this random id and timestamp and store it locally.
    c. On getting tested positive, the user uploads all the random id to a server.
    d. All curious client devices download this data and compare it with their locally stored data.

B. Benefits:
    a. Privacy leakage is very low.

C. Challenges:
    a. Running the transmission (or any other task in general except the entitled ones) in the background is not possible in iOS.
    b. Power consumption could be very high in the case of broadcasting.

**References**

[1] https://wigle.net/index

Other references
https://blog.google/inside-google/company-announcements/apple-and-google-partner-covid-19-contact-tracing-technology/

Contributors:
Ramesh Raskar, MIT and SafePaths
Abhishek Singh, MIT