CryptBNB: Enhancing Location Privacy Using Disposable Identifiers

Ben Mariano, Nirat Saini, and Brook Stacy

I. Abstract

On large wireless networks, users unintentionally reveal their location data, because anyone with access to the network logs can track which access points the user visits over time. With this information, an attacker can learn sensitive information about a user's activities. We introduce a protocol which allows users to opt out of sharing their location information with the network. If the network wants location data from a user, it must ask the user's device for permission. Simulations of users moving through a randomized network showed that even when the number of users is small, it is very difficult for an attacker to uniquely identify user trajectories over time. For example, in a modest network of 10 access points and 100 users, an attacker can correctly guess the full trajectory of a user[1] with probability less than 0.01%

II. Introduction

Users connected to large wireless networks, such as those found at college campuses, shopping malls, or airports, have little location privacy. An honest-but-curious network administrator, or an attacker who has access to the network logs, can track the locations of individual users over time, and can also identify the vendor of the user's device through a device's Media Access Control (MAC) address. The network may collect this information and store it for later processing. A user's location information potentially reveals sensitive data about them. Using this information, an attacker or curious administrator could infer a user's daily schedule, workplace, frequented areas, or other personal information related to location. They could learn which users were collocated at a given time, or even possibly that two users were meeting. This information could be damaging in many contexts.

Currently, there is little that users can do to protect themselves from these threats, and many users may not even be aware that they are being tracked. Using the network automatically makes a user vulnerable to these attacks, and in many cases users do not have the option to avoid using the network entirely. In this paper, we propose a protocol which allows users to opt out of giving their location information to the network. It allows users to scramble their identity in such a way that it is not possible for the network to identify them uniquely. However, users may wish to reveal location information sometimes, or at least they may not care about being tracked at certain times or locations. Our protocol allows users to specify where and when, if ever, they will reveal their identity. At all other times, their identity is obfuscated through the use of disposable

[1] Assuming the user switches MAC address at least 5 times

MAC addresses. If the network wants to learn information about the location of a specific user, it must ask the user's device for permission.

III. Related Work

The field of location privacy is by no means a new one. As wireless technology has grown in popularity, many in the security research community have noted the potential security risks posed to location privacy.

Location Privacy for users with wireless services has been an intriguing topic in the research community. There have been various studies which explore this problem considering multiple use cases and scenarios. Marco et al [1] present an elaborate study on quantitative analysis of enhancing location privacy through frequent disposal of a client's interface identifier. The study generalizes a framework for protecting against an adversary following a user's movements over time. Moreover, it involved design challenges including selection of new interface identifiers, detecting address collisions at the MAC layer, and timing identifier switches to balance network disruptions against privacy protection. As suggested in this study, with large networks, frequent intervals must be considered for MAC switching, so as to avoid identity tracking and collision among identities. Leping Huang et al in their paper [7] talk about silent periods for changing the identifier so that the identities are untraceable among various wireless networking nodes. Their study revealed that silent periods should be constant and variable, to combat correlation attacks for multiple identities in location tracking scenario.
Another study [4] proposes a solution for ensuring location privacy of users in public Wi-Fi networks. This paper strategizes their approach through three steps: 1) anonymize user identity through changing MAC pseudonyms, 2) apply silent periods between different pseudonyms, 3) control power to reduce APs that can register signal. Their system was able to achieve k-anonymity of 1000 in the same-coverage area. While (2) may seem small, we believe that requiring the network to adopt additional computation to support privacy protocols is a sure-fire way of ensuring protocols are never actually implemented. Our protocol will operate without any necessary action of the network provider. We do not plan to specifically address power modulation in our study, as it is out of scope of our threat model (Section IV).

Moreover, Sheng Zhong et al present another approach to deal with location information of mobile devices, by handing three separate component: location service, localizing and communication component. Their study comprises of user getting the control over which entities can have user location information stored, such that third party applications and entities cannot get location of user when connected to any wireless network. This study is mostly constrained to application permissions administration, for avoiding unnecessary applications to get user location via wireless connection. Similar to this, Matthew Lentz et al propose selective recognition and unlinkability of users among peer to peer networks while using specific social platforms in [2]. This paper focuses on the specific problem of communication using mobile social media applications identifying and recognizing nearby devices using short-range radio, which introduces risk for users by revealing their identity to nearby devices without their permission. Moreover, this loss in privacy might be required in the current setup for the user to achieve selective linkability and recognition by closeby devices (such as user's friends). SDDR

(Secure Device Discovery and Recognition) [2] is a bluetooth based protocol which provides secure encounters, while maintaining user privacy in such scenarios.
These studies ([1],[2],[4] and [7]) deal with the problem of location privacy for different wireless technologies, all of them deal with the aspect of hiding the identity of users, with different mechanisms. Once identity is hidden, there is no way to perform any queries on data if required. Hence, privacy as a general term in most of the related work deals with anonymizing the users in any wireless network, and do not talk about partial privacy, with which private user data can be accessed and identified in case user or administrator has to perform any query on the data. One of the common idea discussed when queried architecture is discussed is of CryptDB [3]. CryptDB is a system that allows for queries over encrypted data in relational databases. CryptDB supports a wide variety of queries over the encrypted data which require varying levels of encryption strength. The onion approach is used to update encryption strength on the fly to provide the maximal security possible while allowing the widest breadth of queries. Another interesting study in location privacy for wireless sensor networks discusses location preserving for smart wearables, using low energy bluetooth [6]. Specifically, they propose to use random bluetooth address that can be only recognized by authorized peers or administrators. They also talk about various modes of operation such as opt in/ opt out, k-anonymity and granularity based anonymity, which is somewhat similar to tiered queries we intend to perform with our protocol, for wireless networks.

Most papers and related work in this field consider the problem of location privacy and querying over encrypted dataset as separate problems. The basis of our research proposal is amalgamation of privacy and ability to perform queries over that private dataset, in required circumstances with user permission, which makes the problem unique and unexplored. We have designed a protocol which ensures user location privacy, yet provides user privileges to have control over level of information to be revealed using queries over the private data, if necessary.

IV. Threat Model

We intend to guard against two threats. The first threat is an honest-but-curious network administrator. This adversary has access to the database of all MAC addresses logged by all access points on the network. This network administrator is not malicious, so we assume that they will not apply subversive techniques to the network itself gather more data. A malicious network administrator could make one access point look like a different one if they know that a user is willing to reveal more data at that access point. They could also terminate individual connections to narrow down which users are using which MAC address sets. However, our network administrator adversary is passive and does not alter the data or the network in any way. The second threat is a malicious, passive attacker who has obtained read-only access to the network. This attacker intends to learn sensitive information about users, and can see all of the information gathered by the network, including all logs of MAC addresses. However, they cannot alter the network in any way. We designed our protocol in such a way that an attacker cannot track users even if they have access to all of the MAC address logs gathered by the network. It is important to note that we assume that users are identified only by their MAC addresses. We assume that if an attacker cannot associate these addresses with a user over time, then the user can’t be tracked. In reality, other information, such as active connections or signal strength, may reveal individual users even if their MAC address has changed. However, such threats are outside the scope of this threat model.

Normally, an attacker with access to logs of MAC addresses over time could reveal significant sensitive information about individual users. They could infer where a user works, where they eat lunch, or where they arrive every day. They may even be able to pinpoint where a user works in their specific building. They could do these things by simply tracking the location of the user's MAC address across access points over time. They could also infer that two users were meeting if they both arrive at some area at the same time and leave at the same time. This is especially applicable if the users do this more than once. The attacker may also be able to tell what times a user arrives and leaves from work, when and for how long they eat lunch, or any other data about their daily schedule. There are a multitude of reasons why users may not wish to reveal some or all of this information.

V. Protocol

The CryptBNB protocol provides users locational privacy as they traverse public WiFi networks, allowing users to opt in to varying levels of privacy according to their needs. The protocol makes use of basic cryptographic primitives in order to guarantee anonymity properties. What follows is a description of the tiered privacy scheme, a detailed look at the encryption schemes, and an explanation of queries supported.

V. a. Tiered Privacy

Users can opt in to varying levels of privacy, allowing those who are less security conscious to freely provide locational information to the network while still providing privacy guarantees to more security conscious individuals. The CyptBNB protocol provides four tiers of privacy, described in the table below (Figure 1).

CryptBNB Privacy Tiers	
Name	Description
Don't Care	Freely share true MAC address with network at all times
No Vendor	Only encrypt first 24 bits of MAC address to hide vendor
Spy	Encrypted MAC address that changes every *T*
Limited Spy	User can specify certain time-frames and APs where they do not care

Figure 1: Privacy Tiers

V. b. Encryption Schemes

For No-Vendor, Spy, and Limited Spy, encryption is used to generate new mac addresses. For No-Vendor, the scheme is quite simple. The first 24 bits are encrypted using some cryptographic hash function *f* and some private key *k*. The last 24 bits are left untouched. For Spy-Mode, the scheme is a bit more complicated. The user specifies some amount of time *T*, where *T* is the amount of time in minutes where the device maintains the same MAC address. New MAC addresses are given by the following formula

$$mac = take(48, f(a \mid t \mid k)) \qquad (1)$$

where *take(n, bs)* retrieves the first *n* bits from *bs*, *a* is an access point identifier, *t* is a timestamp, and *k* is a private key. Because *f* is a deterministic function, if we have the device private key *k*, we can compute the mac address a device *would have had* at some location *A* at some time T_1. This ability allows users to answer retroactive queries from the network as to their whereabouts by recomputing the mac address they would have had. For example, if the network wanted to know if a device was at location *X* at time T_2, they can release an active query to connected devices with the information *X* and T_2. If a device wants to respond, they can answer the query by providing the network with the mac address they would have had if they were there (by recomputing *f* on *X* and T_2). The network can verify if they were there by grepping for that mac address.

For Spy-Mode, in order to increase the difficulty of easily connecting a new MAC address to the previous one, devices spoof *M* random mac addresses in time window *w* of a switch, where *M* and *w* are user specified quantities. Mac addresses are generated according to the following algorithm (in pseudocode)

```
if (inWindow($t$, $w$))
   for (i = 0, i < $M$, i++)
      Macs[i] = take(48,  f ($a$ | $T_n$ | $k$))
if ($t$ % $T$ == 0)
   Mac = Macs[rand(0, $M$)]
```

where the array *Macs* contains the spoofed addresses, *Mac* is the device's broadcast MAC address, *inWindow* computes if the current time is within ± *w* of a mac address switch, T_n is the next time t_n s.t. $t_n > t \wedge t_n \bmod T == 0$, and *rand(q, r)* chooses a random integer in [*q*, *r*). One thing to notice is that the device's broadcast MAC address simply assumes the identity of one of the spoofed MAC addresses. The other spoofed MAC addresses will no longer be spoofed once *t* is no longer in-window. Limited-Spy Mode works identically to spy mode, except with an additional check *inDontCare*(*t*, *a*) that checks if the time *t* and/or access point *a* are within the user-specified don't care times/regions.

V. c. Queries Supported

Any query is trivially supported over users who opt for Don't Care. In order for a network to know a device is in Don't Care mode, a user must notify the network. If a user does not do this, a smart network could use heuristic metrics (i.e. total time MAC has been on network), in order to determine if a MAC address belongs to a Don't Care user. In a similar fashion, any non-vendor query is supported for users in No-Vendor mode. No *unassisted* queries are supported over Spy users, where *unassisted* refers to queries run without additional information from the user. However, *assisted* queries are supported for Spy users. Assisted queries come in the form of two *base* queries, from which many other queries can be derived. The two base queries are: 1) From what vendor is device *X*? 2) Was device *X* connected to access point *B* during the time period [t_1 - t_2]? If a network desires an answer to either of these two questions, it can query current active devices on the network with these questions. Question 1 can be trivially answered by the device if the user so wishes. Question 2 can be answered by calculating the MAC address/addresses that the device *would have had* if it was at access point *B* during time period [t_1 - t_2]. Because the MAC address generator is only a function of time, location, and secret key by (1), the MAC address/addresses can easily be recomputed. While Question 2 may seem trivial, many interesting queries can be derived from it, such as, *Were devices X and Y co-located at time t somewhere on the network?*, *Did device X connect to access point A in time period [t1-t2]?*, *Where was device X at time t* and similar queries.

VI. Simulation and Evaluation

To test our protocol, we used simulations to experimentally verify our privacy claims. In order to simulate our protocol, we simulated graphs of access points and users who traverse that graph. Users enter and exit the graph at time intervals of 10 minutes. Below is an in depth description of our simulation.

VI. a. Graph Simulation

To simulate a network for the users to traverse, we used a graph generation algorithm. In this graph, nodes represent groups of access points within communication range of each other, and edges represent a path from one group to another. We required this graph to have several properties in order to more closely emulate a real-world network. First, for simplicity, it should be connected. We believe this is an acceptable condition because generally, in real networks, paths exist between all areas. It should also not be uniform. Instead, it should have some dense areas and some simple paths connecting them, because real networks have buildings or other hubs and also roads or paths connecting them. It is also important that this network has a realistic two-dimensional embedding. To generate a network satisfying these conditions, the following algorithm was used.

First, choose n random points on a 2D plane, where n is an input. These will be the nodes of the graph. There are n*(n-1)/2 possible edges in this graph. The length of an edge is the euclidean distance between its two nodes on the plane. We first compute the minimum spanning tree of these nodes and add its n-1 edges to the graph. This ensures that the resulting graph is connected. Next, we consider k, the density value. This is a user input which is a decimal value between 0 and 1, and it is used to adjust the density of the resulting network. In the end, the graph should have (n*(n-1)/2)*k total edges, though we enforce a minimum of n-1 edges for the spanning tree. If k=0, a tree would be produced. If k=1, the graph would be fully connected. After adding the spanning tree, we also add (n*(n-1)/2)*k – (n-1) more edges, in order from shortest to longest, skipping edges which have already been included. Now, we have the correct number of edges.

This algorithm allows the size and density of the graph to be adjusted. Because it starts as a set of points in 2D space and prioritizes adding the shortest edges, it always has a sensible 2D embedding. It also ensures that less dense graphs are always subsets of denser graphs. It is important to note that choice of an appropriate k depends greatly on the choice of n, because the number of edges increases quadratically with the number of nodes. For example, k = 0.5 may be reasonable for n = 10, but it would produce an extremely dense graph for n = 1000. Below, we have included some example graphs of size 100.

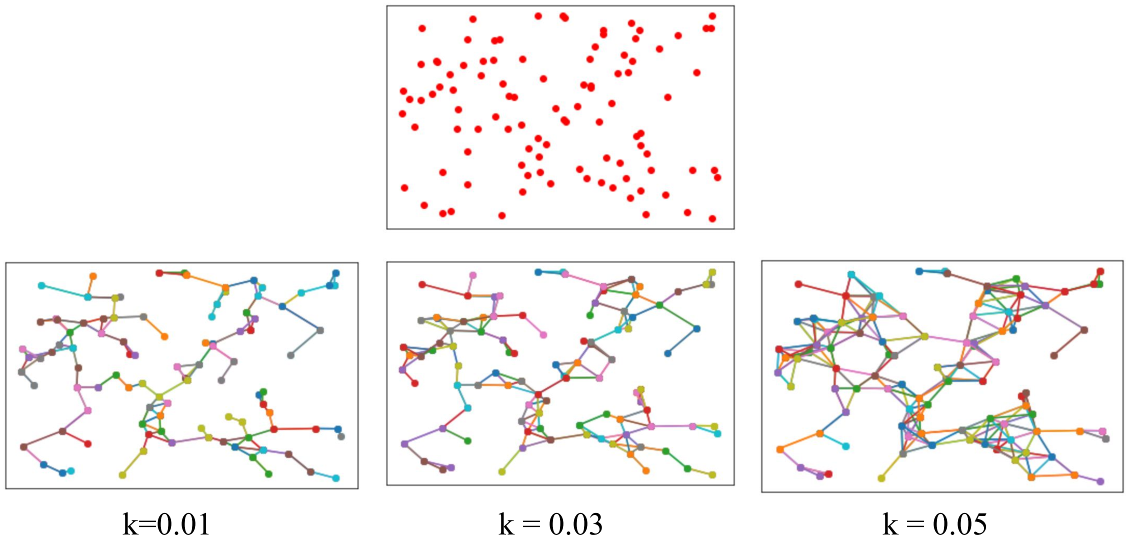

Figure 2: Topology graphs of access points with different degree of connectivity

VI.b Protocol Simulation

The protocol simulation attempted to quantify some of the privacy properties of the protocol. Specifically, we aimed to answer the following questions: 1) Are MAC address collisions a problem as identified by [4]? 2) How does network population density affect privacy? 3) How many people must participate in order to achieve reasonable privacy guarantees?

Tao Jiang et al. take specific actions in their privacy protocol to avoid MAC address collisions, forcing the access point to assign random MAC addresses [4]. However, this approach is undesirable for two reasons: 1) This requires the network provider to buy into the protocol in order for anyone to benefit from it 2) This makes it much more difficult to become anonymous to the network if the network provides MACs. For these reasons, we opted to remove AP MAC provision in our protocol, at the potential expense of MAC address collisions. In a span of over 1000 tests using varying graph sizes, populations, and user settings, we found that we had no instances of simultaneous MAC address collisions, meaning that two devices had (or spoofed) the same address at the same time. This is expected given that the probability of a mac address collision given *n* randomly distributed MAC addresses is given by the following

$$p(n) = 1 - \frac{m * (m-1) * \ldots * ((m+1)-n)}{m^n} \qquad (2)$$

where *m* is the space of MAC addresses (2^48). Using this calculation, even with 1 million devices at an AP, one expects a collision with only 0.17% likelihood.

In order to quantify locational privacy, we use the minimum, average, and median number of devices a switching device could be matched with. If a MAC address disappears at timestep *t*, there are two possibilities: the device left the network or the device switched to a new MAC address. If the attack assumes the device switched, the number of potential devices it could have switched to is the number of devices *n* that coexisted at the same AP at timestep *t*. A slightly smarter adversary may make the heuristic guess that he can subtract the number of devices *r* that have been on the network for at least *l*, where the attacker assumes $l > w_{max}$ where w_{max} is the maximum of all window sizes for users spoofing on the network. Assuming the attacker can perfectly predict whether a device left or switched, the attacker can guess the connection between the old and new MAC addresses with probability 1/*n*. One important note is that the protocol ensures *n* must be greater than or equal to *M*, the number of spoofed MAC addresses by a user. Additionally, assuming a user *U* is on the network for *N* switches of her MAC address, the probability that an attacker can guess the user's entire trajectory is

$$\frac{1}{(n_1-r_1)*(n_2-r_2)*\ldots*(n_N-r_N)} \qquad (3)$$

where n_i is the number of devices that coexisted at the same AP as *U* prior to switch *i* (r_i follows from the description above).

In order to quantify the effect of population density, we ran tests varying the number of nodes and population of users on the network. The number of nodes varied between 5 and 100 (in increments of 5) and the population between 100 and 10000 (in increments of 100). For this experiment, all users were considered to be in Spy-Mode. This assumption was made in order to control for changes due to variations in user preferences and reducing the size of testing. This assumption means all results are a ceiling. Additionally, we assumed the network made the heuristic guess that *l* is 30 minutes; that is that any device that has been present on the network for more than 30 minutes is not considered as a potential candidate. This heuristic was chosen as 30 minutes would be a very long time to spoof MAC addresses, but not so long as to discount very privacy aware users. Experimenting with varying heuristics here is an important area of future work, as it could have impacts on anonymity achieved. Our results were as follows:

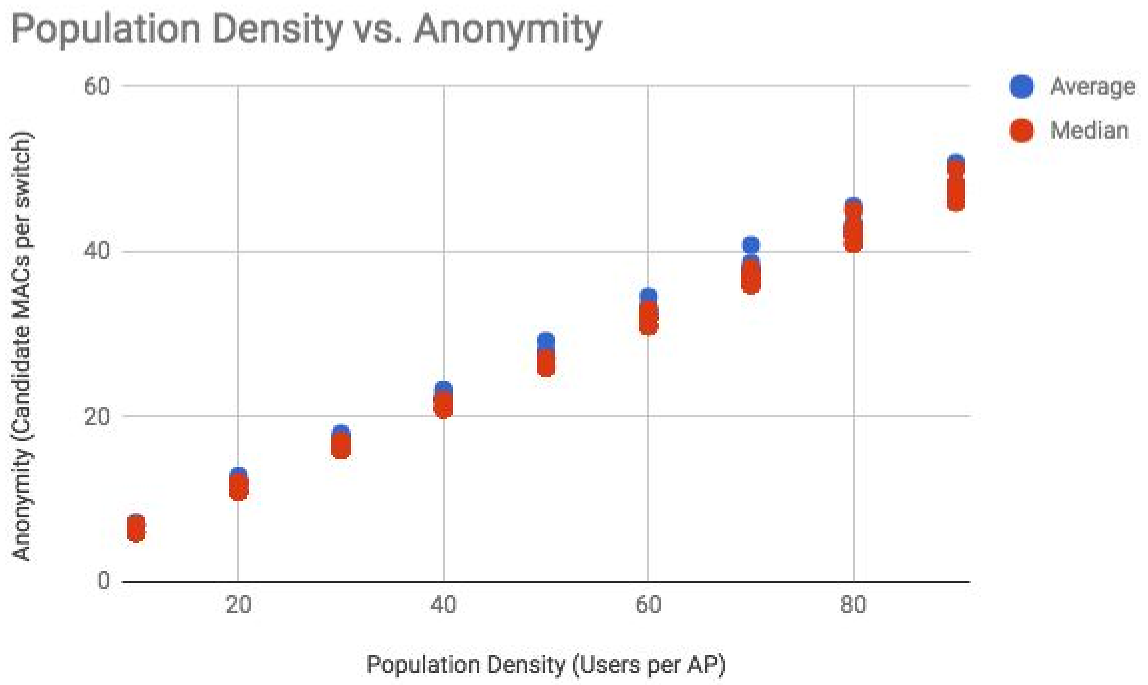

Figure 3: Population Density versus Average/Median Anonymity

As shown above, the median and average candidate MAC addresses were very close to one another. For the lowest densities (10 people per access point), the average candidate choices per switch was about 6, while for the highest densities (100 people per access point), the average candidate choices per switch was about 48. To put these numbers in perspective, these findings show that even in low density areas, an attacker can guess the full trajectory of a user who switches MAC addresses 5 times with probability 0.01%.

Finally, we attempted to determine how many users must participate (i.e. Full or Limited Spy Mode) in order to ensure different levels of privacy. In order to determine this, we used a constant population density, and only varied the percentage of users in Don't Care, Limited, and Full Spy. We assume for the sake of this experiment that no users are Vendor-Only, nor are there any Limited-Vendor-Only. Our results were as follows

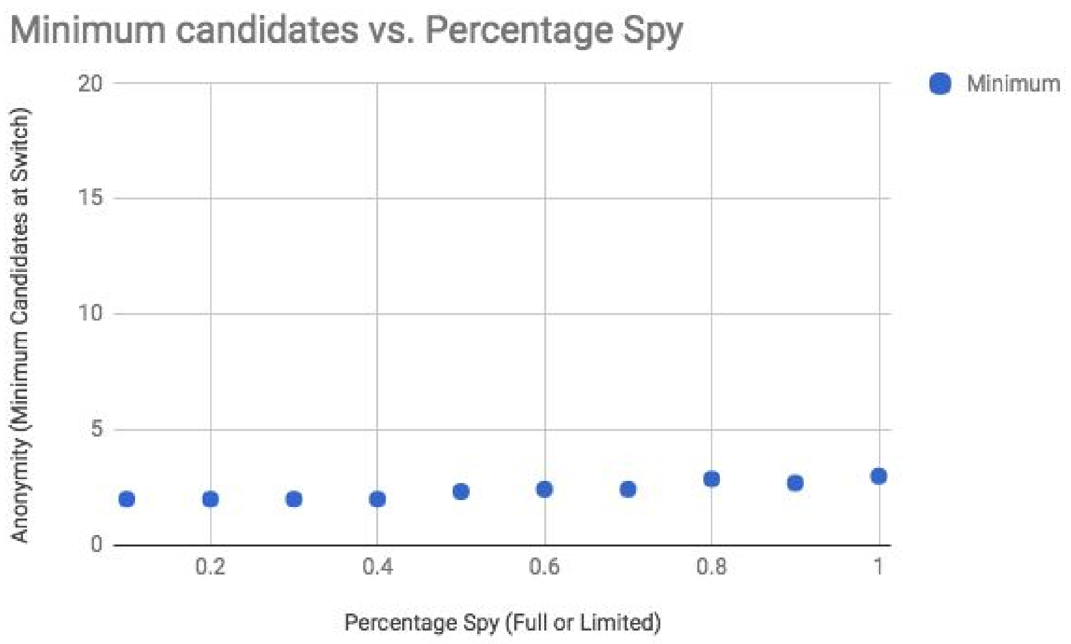

Figure 4: Minimum Candidate MAC Addresses at a Switch for Varying Percentages of Spy Mode (Full or Limited) Devices

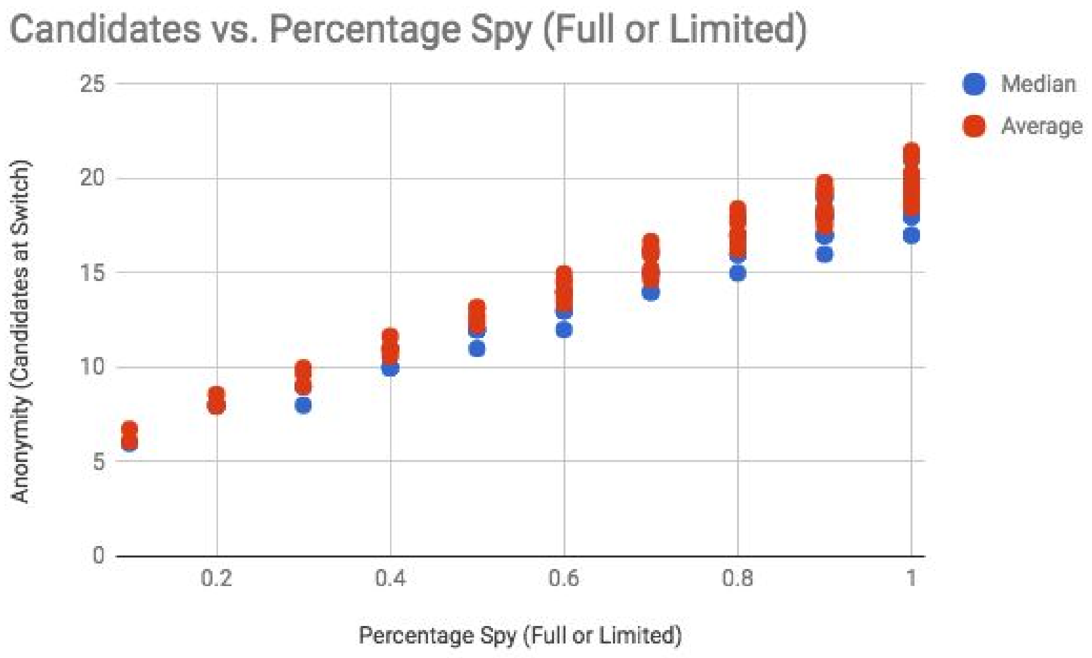

Figure 5: Average and Median Candidates at a Switch for Varying Percentages of Spy Mode (Full or Limited) Devices

As shown by the graphs above, varying the percentage of users participating has little effect on the minimum number of candidates on a switch. In many cases, the minimum was limited to the number of spoofed MACs (2 for this experiment). However, the average and median candidate devices increased with percentage participating, as one might expect. The most promising result from this is that even with only 10 percent of users participating, they have an average candidate number of around 6 devices. These results mean that the protocol is very effective for users who wish to hide whole trajectories from the network (0.01% for a user with 10% participation with 5 changes), but works less well for users wishing to hide single MAC address changes (as bad as 50% for single switches). One way to improve the hiding of single switches is to increase the number of spoofed MAC addresses, as this is the absolute minimum number of candidates from which a user can be identified.

VII. Security Analysis

In this section, we discuss the security of our proposed algorithm, CryptBNB, with respect to various attacks. Moreover, we talk about unique properties of CryptBNB which ensure security from various attacks both in and out of our threat model.

In order to define our attacks, we should first remind the reader of our notion of anonymity. In our study, anonymity is twofold: 1) An individual should not be able to be linked to a MAC address 2) When a user switches from one MAC address to another, an attacker should not be able to link the two addresses together (i.e. track a user's trajectory through time).

VII. a. Simple Elimination Attack

Using CryptBNB, each user changes their identity from one pseudonym to another. In this process, to ensure that the attacker cannot track the new identity of a user, each user associates itself with Fake MACs. In this way, the attacker cannot perform a passive attack with network log, to link or backtrack the pseudonym changes for each user at a single access point and even over the network.

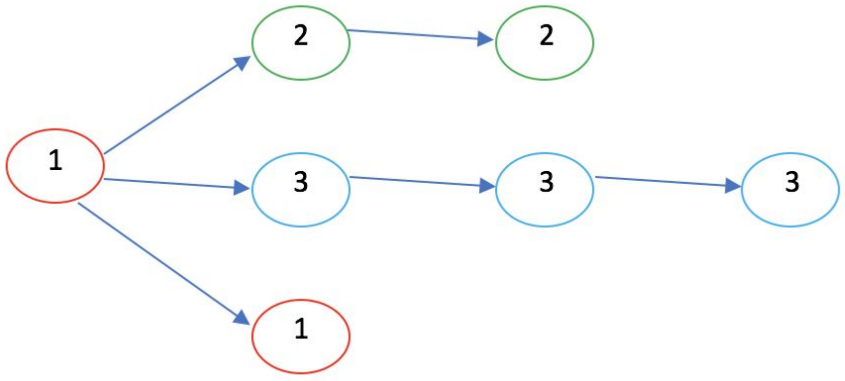

Figure 6: Transition from one pseudorandom to another

Assume user *X* currently has a pseudonym MAC, represented by 1 at time t_0. *X* wants to remain anonymous such that while switching from one MAC to other, attacker cannot map the changed MAC for *X*. To achieve this, at time t_1, *X* introduces two more identities, faking MACs 2 and 3 in the network along with 1. At t_2, original identity 1 disappears, still *X* acts as identities 3 and 2. At the next time unit t_3, identity 1 disappears and user *X* successfully changes the pseudonym MAC, being anonymous and avoiding any back-tracking. Even if attacker is actively observing the MAC address changes on an access point, it is hard to infer that user *X* switched from identity 1 to identity 3, due to involvement of identity 2. Even considering worst case, an attacker can at most infer user X to be one of the identities 3 or 2, which can be asserted with probability 0.5. If the user spoofs more number of entities at once, not just 2 as described in this example, the probability of tracking a user becomes even lower. The technique of spoofing MACs to avoid simple elimination attacks is a unique property of CryptBNB.

VII. b. Device Inference Attacks

CryptBNB uses a cryptographic hash function (MD5, SHA-1, etc.) to encrypt a user's MAC address (or only the first 24 bits if in No-Vendor Mode). Using the avalanche effect property of hash functions, even a small change in the data *d* leads a large difference in hashed data *h(d)*. Therefore, even users who have the same vendor (i.e. Apple) should have substantially different MAC addresses. Similarly, even if an attacker knows a given user's real MAC address, due to the avalanche effect, they have a very small probability of determining the user's generated fake MAC addresses. Additionally, because the access point and time are used as arguments to the hash function, even if an attacker had access to a devices secret key (used for hashing), they still could not recreate a user's associated MAC addresses without knowing the times and access points the user was at. If the attacker already knows the times and access points, then the location privacy is unnecessary. If they do not know this information, in smaller networks of relatively few APs, a patient attacker could run a brute force attack, calculating all MAC addresses a user could have for every AP at every time of day.

VII. c. User Inference Attacks

User inference attacks leverage external knowledge of users in order to deanonymize data. These attacks are outside of our threat model, but are important and realistic attacks for certain users. For example, if an attacker knows that a certain user *X* was at *A* at 8:00, *B* at 9:00, and *C* at 10:00, they could use this information to identify a MAC address *M* with the same trajectory. If there exists only one mac address *M* with this trajectory, then the attacker can conclude that *M* is *X*. However, if users can spoof MAC addresses, the attacker can no longer conclude that *M* is *X* in this case, as *X* could have spoofed many MAC addresses, and therefore *M* could be a different user. If an attacker knew that *X* switch from *A* to *B* at exactly 8:43, they have a much better chance of identifying *X*'s spoofed MAC address *M*, as the number of users who makes this switch at exactly this time could be fairly low (depending on the population of the network and the likelihood of a user making this transition).

In a similar manner, an attacker might be able to leverage known user-specific use patterns in order to deanonymize the data. For example, if a user *X* is connected to a specific AP *A* on a fixed schedule, an attacker might be able to determine the MAC address *M* that *X* has at that fixed schedule. However, due to MAC switching and the ineffectiveness of guessing at switches, an attacker would not be able to track the user for long trajectories outside of the known schedule. This means that even if an attacker knew that *X* has MAC address *M* at time *t* due to user-specific behaviour, their ability to determine *X*'s trajectory up until time $t+\Delta$ is

$$\frac{1}{n_p * n_{p-1} * ... * n_1} \quad (4)$$

where n_i is the number of devices that coexisted at the same AP as *X* prior to switch *i* and *p* is the number of MAC address switches in time period Δ.

Another user inference attack is linking two devices: one who is in Don't Care Mode (doesn't care about revealing their location and private information) and another who is in Full Spy Mode (encrypts entire MAC address at all times). If an attacker knows (via external knowledge) that two devices like this are linked, they could use this to deanonymize the second device who wishes to remain anonymous. If the two users are always collocated, the Spy-Mode user's anonymity is completely eliminated. However, at any time when the users are not collocated, the Spy-Mode user enjoys the anonymity described by (4). While a *co-linkability* attack of this sort may be unlikely for two different users with devices, it is a plausible attack for one user who has two different devices, one of which is not running CryptBNB.

VIII. Future Work

CryptBNB works at the level of MAC address stored at access point logs and does not consider packet level information. An attacker can still perform deep packet inspection to know the applications used, to backtrack the series of pseudonyms used by any specific spy-mode enabled user. Right now, our protocol does not defend against such attacks. One area of future research is figuring out if it is feasible to obscure identifying packet information in order to maintain the anonymity provided by CryptBNB. Additionally, CryptBNB does not account for signal strength as an identifying characteristic. Future work would include combining CryptBNB with ideas from Tao Jiang et al. [4] for dealing with signal strength. Another potential weakness of the protocol is the fact that randomized MAC addresses have completely random vendor identifiers (first 24 bits). This could make fake MAC addresses easily identifiable if only a smaller subset of the vendor space is used by real devices. Future work would include studies of the distribution of device MAC address vendor specifiers. CryptBNB is also limited by practical limitations. Particularly, switching and spoofing MAC addresses over an open connection are a concern. For instance, if a user switches MAC addresses, what effect will this have on the connection? Similarly, is it feasible for a single Wi-Fi card to realistically spoof multiple simultaneous MAC addresses? One potential solution for this would be a smarter protocol for switching MACs in periods of device downtimes (i.e. if the device screen is off, no user activity for 5 minutes, etc.). Exploring the practical limitations are perhaps the most important area of future work, as the usability of this protocol hinges on its feasibility on real networks.

IX. Conclusion

CryptBNB provides location privacy for users on public Wi-Fi networks as they need it, while still enabling the networks to query important information from users. With simulation experiments, we observe that even with small numbers of participating users, CryptBNB can provide reasonable anonymity from the network administrator. With a user-centric query design, the CryptBNB protocol need not have the network participate in order for users to enjoy location privacy. In a departure from other solutions, CryptBNB still supports queries over users who allow it, and even allows for retroactive queries for important information such as collocational information for tracking disease outbreak. CryptBNB provides a novel solution for locational privacy on public networks.

X. References

[1] Marco Gruteser and Dirk Grunwald. 2005. Enhancing location privacy in wireless LAN through disposable interface identifiers: a quantitative analysis. *Mob. Netw. Appl.* 10, 3 (June 2005), 315-325. DOI=http://dx.doi.org/10.1007/s11036-005-6425-1

[2] Matthew Lentz, Viktor Erdélyi, Paarijaat Aditya, Elaine Shi, Bobby Bhattacharjee, and Peter Druschel. 2014. SDDR: light-weight, secure mobile encounters. In *Proceedings of the 23rd USENIX conference on Security Symposium* (SEC'14). USENIX Association, Berkeley, CA, USA, 925-940.

[3] Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. 2011. CryptDB: protecting confidentiality with encrypted query processing. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles* (SOSP '11). ACM, New York, NY, USA, 85-100. DOI=http://dx.doi.org/10.1145/2043556.2043566

[4] Tao Jiang, Helen J. Wang, and Yih-Chun Hu. 2007. Preserving location privacy in wireless lans. In *Proceedings of the 5th international conference on Mobile systems, applications and services*(MobiSys '07). ACM, New York, NY, USA, 246-257. DOI=http://dx.doi.org/10.1145/1247660.1247689

[5] Sheng Zhong, Li Li, Yanbin Grace Liu, and Yang Richard Yang. Privacy-preserving location- based services for mobile users in wireless networks. *Technical Report,* Yale Computer Science, July 2004.

[6] Zhuo Chen, Haibo Hu, and Jielin Yu, Privacy-preserving large-scale location monitoring using bluetooth low energy, in *2015 11th International Conference on Mobile Ad-hoc and Sensor Networks (MSN)* (2015), pp. 69–78. doi:10.1109/MSN.2015.38

[7] Leping Huang, Kanta Matsuura, Hiroshi Yamane, and Kaoru Sezaki. Enhancing wireless location privacy using silent period. *In Proceedings of IEEE Wireless Communications and Networking Conference* (WCNC), New Orleans, LA, 2005.

[8] Aditya Jitta and Arto Klami. Partially hidden Markov models for privacy-preserving modeling of indoor trajectories. In *Neurocomputing Volume 266,* 29 November 2017, 196-205.

[9] Lorenzo Bergamini, Luca Becchetti, and Andrea Vitaletti. Privacy-Preserving Environment Monitoring in Networks of Mobile Devices. In *International IFIP TC 6 Workshops*, 13 May 2011, 179-191.

[10] Abdulatif Alabdulatif, Heshan Kumarage, Ibrahim Khalil, and Xun Yi. Privacy-Preserving Anomaly Detection i n Cloud with Lightweight Homomorphic Encryption. In *Journal of Computer and System Sciences Volume 90,* December 2017, 28-45.

[11] Michael Kamp, Christine Kopp, Michael Mock, Mario Boley, Michael May. Privacy-Preserving Mobility Monitoring Using Sketches of Stationary Sensor Readings. In *Machine Learning and Knowledge Discovery in Databases*. ECML PKDD, 2013.