IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 23, 2020, accepted February 15, 2020, date of publication February 24, 2020, date of current version March 4, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2976189

Detection of Spoofing Attacks in WLAN-Based Positioning Systems Using WiFi Hotspot Tags

AYONG YE, (Member, IEEE), QING LI, QIANG ZHANG, AND BAORONG CHENG
Fujian Provincial Key Laboratory of Network Security and Cryptology, Mathematics and Informatics Department, Fujian Normal University, Fuzhou 350007, China

Corresponding author: Ayong Ye (yay@fjnu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61972096, Grant 61872088, Grant 61872090, and Grant U1905211, and in part by the Natural Science Foundation of Fujian Province, China, under Grant 2018J01780.

ABSTRACT WLAN-based localization is widely adopted for mobile positioning, which is the prerequisite for location based services and more. However, the existing positioning systems are vulnerable to location spoofing attacks, which may bring major privacy concerns to mobile social network service (MSNS). In this paper, we first show a privacy attack model base on spoofing attack in MSNS, and then propose a novel defense mechanism based on WiFi-hotspot tags (i.e. base-station tags, BS tags). Specially, we utilize the unpredictability and reproducibility of BS tags to authenticate the spatial-temporal property of a geolocation. Furthermore, we introduce the bloom filter to compress parts of real-time hotspots frames, while guaranteeing its high entropy. Also, we design a tag verification algorithm based on the fuzzy extractors, which can well adapt to the high-bit error rates of wireless transmission. Finally, the safety and feasibility of the mechanism are proved by theoretical and experimental analysis.

INDEX TERMS Social network, WLAN-based positioning, location spoofing attack, WiFi hotspot tags, fuzzy extractors.

I. INTRODUCTION

The proliferation of smartphones and ubiquitous Internet have given rise to mobile social network service (MSNS), which has become an important part of people's daily life [1]. MSNS is a combination of traditional location-based service (LBS) and online social network service. People can enjoy many application services in MSNS such as Foursquare and Sina Weibo [2]. Such social applications provide various kinds of personalized services combined with location information, which let users post and share information about themselves, and connect with others sharing their informations via mobile.

Apparently, positioning technology is the prerequisite of the location based MSNS, which has been achieved either through the use of satellite, inertial sensor and WiFi network [3]. In comparison of the first two techniques, WLAN-based positioning is seen as a more advantageous option in many occasions thanks for its good balance of cost and coverage. However, WLAN-based positioning systems are susceptible to various malicious attacks owing to the openness of wireless networks. One of the major security concerns is that the reported locations could be easily forged by malicious users in order to exploit the benefits of proximity detection service. For example, Tippenhauer *et al.* [4] demonstrated that Skyhook positioning system is vulnerable to location spoofing and location database manipulation attacks, using Apple's iPod touch and iPhone as the positioning devices, which the attacker could arbitrarily change the localization information at the victim device.

Since vulnerable to location spoofing attack, MSNS users are suffering severe risk of privacy leakage due to the social network sharing and location relation. In this paper, we defense location spoofing attack in MSNS by proposing an BS tags based location verification mechanism. A BS tag can be regarded as a token of proof associated with a geolocation near a BS during certain time, which is generated based on the random frame signals sent by WiFi hotspots. Then, we utilize the unpredictability and reproducibility of BS tags to authenticate location information.

Thus, our contributions are summarized as follows:

(1) To our knowledge, this paper is the first reported research work in literature that introduces the notion of BS tags to detect forged BS or position-changed BSs, while

The associate editor coordinating the review of this manuscript and approving it for publication was Zhenhui Yuan.

preventing the spoofing attacks in WLAN based positioning system and guaranteeing MSNS user privacy. Formally, we provide a verification mechanism to authenticate both spatial and temporal property of a geolocation, which is based on the unpredictability and reproducibility of the BS tags.

(2) We show a privacy attack model base on location spoofing attack in MSNS. In this model, we first mislead the mobile device to locate itself on a specific false position though a location spoofing attack, and then identify the user's real ID according to her specific location.

(3) We compress the real-time signal of BS frames into the BS tags with a bloom filters, while reducing storage overhead of the application systems and avoiding the mapping conflicts. Furthermore, we discretize the time dimension of BS to reduce the communication overhead of the application system, while ensuring the feasibility our defense mechanism.

(4) We also introduce the notion of fuzzy extractor (FE) to improve the robustness of our defense mechanism. Due to the fact that there are high bit error rates in the transmission of wireless signal, the BS tags generated by different users at the same time and location may be not exactly identical. Thus, we utilize the fault tolerance of FE to reconstruct the same key form those inconsistent BS tags.

The rest of the paper is organized as follows. The related work is presented in Section II. Section III introduces the method of privacy attack based on location spoofing. In section IV, a defense mechanism utilizing BS tags is proposed. Section V presents the experimental analysis on the sources of BS frames, computational, storage and communication overhead and the analysis on T (the time period for generating BS tags). Section VI presents the security analysis. Finally, we conclude our work in Section VII.

II. RELATED WORK

In recent years, WLAN-based location estimation applications have been extensively investigated in the context of LBS. Many security analyses had been given to show that these systems are vulnerable to various spoofing attacks, which can be IP spoofing, MAC Spoofing and replay attack etc [4]–[6]. Specifically, MAC spoofing is a method for changing the Media Access Control address (MAC address) of a network interface on a device.

It is worth mentioning that Spoofing attackers can disguise themselves as other legitimate users, thereby positioning themselves into a particular location to obtain special services. Furthermore, the sharing of position information promotes users' interactions in MSNS, while increasing the potential risks of the users' personal information exposure at the same time. For instance, Henne *et al.* pointed out that it is possible for an attacker to speculate a user's position through the geo-tagged social network media information in the WiFi environment, such as the photos appended position characteristic [7]. In [8], Cho *et al* demonstrate that a different type of indoor positioning system using high-frequency audio signals can also be vulnerable to similar location spoofing attacks, through a case study: an in-depth security analysis of the recently launched Starbucks service called Siren Order. Cunche [9] fabricated the access point (*i.e*, AP, or BS) that the target user has previously connected to via the WiFi replay attack, and tracked the user's web data to obtain the user's private information. Celestin *et al.* [10] set up a WiFi network in a special position and conducted the positioning attack. They established the relationship between the target device and the attributes of users by the side-channel attack. This paper will propose a privacy attack that misleads the device to locate to the specific position through the fake BSs, and then utilizes social network sharing and the uniqueness of the location to obtain the user's identity information.

To mitigate localization spoofing attacks, Zhang *et al.* in [11] proposed the Secure Location of Things (SLOT) framework to mitigate the spoofing attack. They reformulate the location estimation problem as a statistical nonlinear estimation model, and then proposed two algorithms based on a mixture model and a time-difference-of-arrival to calculate the MLE (Maximum Likelihood Estimation) for the tags location. In [12], [13], the location tag is proposed to address the threat of location spoofing attacks, which is constructed by location information related to characteristic parameters, such as noisy location data from radio frequency signal, data from wireless AP frame or channel data from GSM base station. Zheng *et al.* [14], [15] also proposed a location based handshake and private proximity test protocol based on spatial-temporal location tags. Akinboro *et al.* [16] implemented an enhanced steganography adaptive neuro-fuzzy algorithm for securing the ambient home network against spoofing attacks, which comprises image steganography, adaptive neuro-fuzzy and transposition cipher. Against MAC and IP spoofing attacks, Yu *et al.* [17] proposed a proximity-based access control scheme to monitor a device's physical characteristics and check if there are any significant changes in the measurements of those characteristics to detect an unauthorized device. At the same time, they analyzed how the RSSI values of packets are affected with changing physical distances in a real network environment.

Compared with current state-of-the-art methods, we compress the real-time frames of a BS over a period of time into a BS tag, which be regarded as a token of proof associated with the nearby environment of a BS. Furthermore, we provide a mechanism to authenticate both spatial and temporal property of a geolocation, which is based on the unpredictability and reproducibility of the WiFi hotspot tags.

III. PRIVACY ATTACK BASED ON LOCATION SPOOFING ATTACKS

A. LOCATION SPOOFING ATTACK

WLAN-based localization is the mainstream technology for positioning smartphone. It is known to us that each BS has a globally physical address (e.t. MAC address) and is fixed at a position during a period of time. Therefore, smartphone positioning systems, such as Skyhook [4] or Google positioning systems, usually locate devices using Wi-Fi and Cell signals. The basic principle of WLAN-based localization technology can be briefly described as follows. In the initialization phase, the location service providers try to collect all BSs' location in advance. In the positioning phase, the location servers search the corresponding reference information of those BSs near mobile terminals in their database, such as the BS's MAC address and signal strength, and then compute the device's position based on signal fingerprint or the trilateral measurement 18].

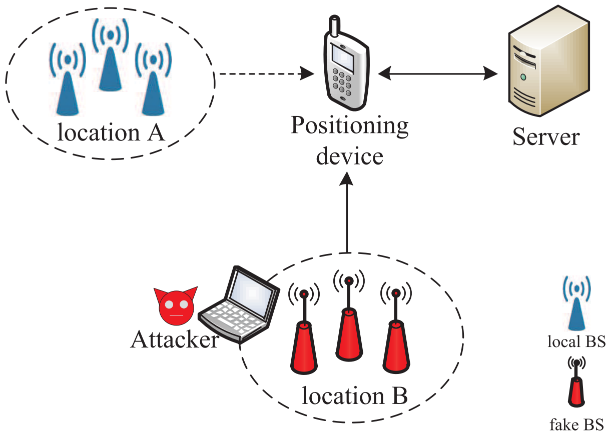


FIGURE 1. The schematic of the location spoofing attack model. The attacker can mislead the target device into location B by forging BSs, while the target is actually in location A.

The attacker can mislead the target device to specific place by forging BSs as the server doesn't verify the validity of BS information uploaded by users. The basic idea of the location spoofing attack model is illustrated in Fig. 1. In the model, the attacker completes the location spoofing attack by interfering with normal wireless communication. To be specific, the attacker forges the BSs located in place B, which is close to the target user in location A. The target user uploads the BS list including fake BSs information to the server for requesting location-based services. If the signals of fake BSs are stronger than those of the local BSs, the server calculates the position of the positioning device mainly based on the counterfeit BSs, which will transmit the wrong location B as the positioning result to the user.

The steps of the location spoofing attack model are detailed as follows.

Step 1 (Choosing a Target Location for Spoofing):

In order to increase the success rate of location spoofing attack, the number of fake BSs must be more than that of actual ones, which ensures that fake BSs are the main references of positioning in target area. In the other word, the target location chosen for spoofing should near as many BSs as possible.

Step 2 (Collecting the List of BSs in the Target Location):

After determining the target location, attackers need to collect all nearby BS's information including SISD and BSSID (*i.e*, MAC address). The easiest way is to search from a public location server, such as WiGLE [19] or Skyhook, which had recorded more than billion BSs in the world. In addition, she/he can scan the Wi-Fi signals near the target location though social engineering.

Step 3 (Weaken Local BSs Signals):

WLAN-based localization is mainly calculated by BS signal strengths and BSSID. The adversary can utilize special software such as USRP Rev. 4.2 [20] or MDK3 [21] to jam the wireless channels. In principle, if the original BS signals are completely shielded, what the local terminal scans are all the fake BS signals, and the user must be localized to the target location. However, in order to prevent the suspicion caused by the paralysis of the local wireless network, the attacker only needs to weaken the intensity of the native BS signals properly. In most cases, jamming real BSs is not necessary and thus the attack could proceed unnoticed by other users.

Step 4 (Forging the Target Location's BSs):

There are many wireless penetration tools under Linus system, such as Aircrack-ng [22] or MDK3, can easily forge BS' wireless frames. It is worth mentioning that rogue BS dispense with the full functions except broadcasting Beacon and BSSID. Intuitively, the rogue BS should be deployed as close to victim as possible to guarantee the success of the spoofing attack. Since positioning system usually picks those BSs with the stronger signal as positioning references.

In our attack setup, the legitimate BSs are transmitting on WLAN channels 1, 6, and 11 (up to 13 channels are available in 802.11b/g). The attacker needs to broadcast the forged BSs intensively in these three channels, which can block the communication between BSs and the terminal. Not to jam the local BSs, the fake BSs can be broadcasted in other channels.

As a motivating example shown in Fig. 3, we conduct a simulative location spoofing attack to our device in Fuzhou city, by forging some BSs located in Shanghai city. The statistical result regarding the success rate of the location spoofing attack is showed in Fig. 2, which is based on 2,600 experimental observations. It is obvious to find that the success

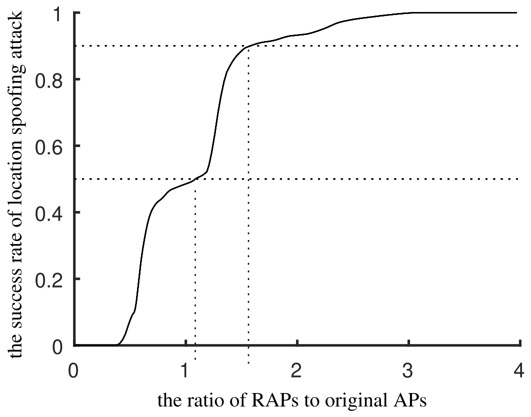
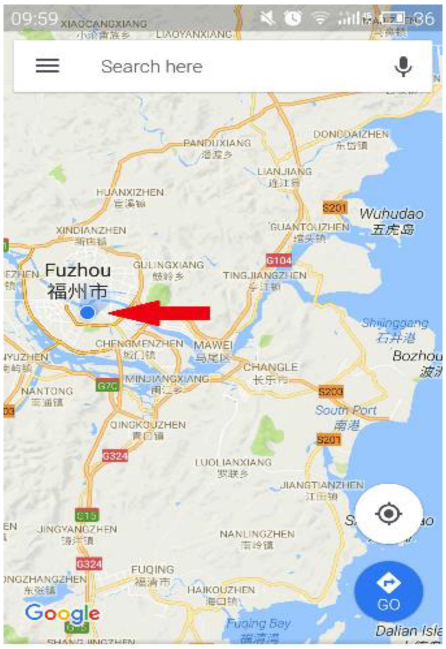
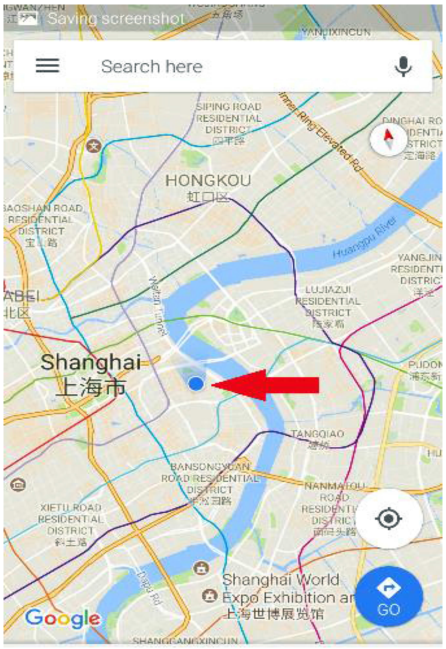


FIGURE 2. Fraction of successful attack as a function of the fake BS ratio. With the increase of fake BSs, the attack's success rate raises. If the number of fake BSs is more than 1.5 times of the local's, the success rate is above 90%.



(a)



(b)

FIGURE 3. The Google Map localization results: (a) is the original positioning in Fuzhou, (b) is positioning results under location spoofing attacks.

rate of the attack increases with the increase of fake BSs. Furthermore, the attack hardly succeed in the case that the number of fake BSs is less than that of the original BSs.

B. PRIVACY ATTACK

Location is the bridge between cyberspace and the real world. Thus, if deceived by location spoofing, a MSNS user may expose their identity (ID) when sharing information combined with location information. The principle of privacy attack based on location spoofing in MSNS is described as follows. To obtain the target user's ID, the attacker would need to deceive the user to locate to a special position by using fake BSs, monitor the social network data shared by all users and then identify the target using the uniqueness of the position.

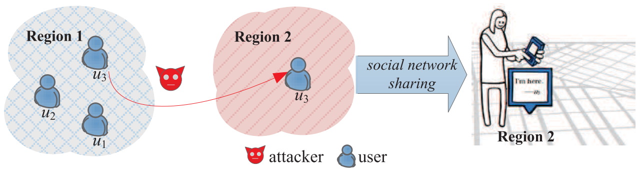


FIGURE 4. The privacy attack based on location spoofing attack.

As shown in Fig. 4, users $u_1 \sim u_3$ and the attacker are in the region 1, but there is no user in the target place (region 2). For instance, the attacker selects u_3 as the target and misguides she to locate in the region 2. Then, the attacker monitors and collects the social network data related to region 2. Since there are no other users in the region 2, the collected information can only be shared by u_3, which can determine the ID of u_3. If there are other users in the region 2, the attacker can record their ID previously, and the known ID should be removed from collected data after the attack. So the attacker can determine whether a user in the objective world is associated with her digital identity, and then the attacker can identify the user and acquire other private information by further cross comparison.

Based on the above analyses, we implement the WLAN-based location spoofing attack on some android applications. Combined with features of the MSNS, we also implement the privacy attack. The results are showed in TABLE 1. It includes four kinds of seven common applications related to positioning, including traffic navigation, social networking, browsers, and mobile payment. Among them, social networking is mainly used to study security potential threats caused by privacy attacks. As shown in TABLE 1, these applications will be located to wrong place due to location spoofing attacks. Correspondingly, location-related functions will send some error message pushes. Among the mentioned four social networking applications, social network information such as nicknames, digital identities, photos, and social relationships of Twitter and Sina Weibo users can be obtained by the attacker, and the security of QQ is relatively high. It can be seen that privacy attacks can cause users' social network privacy to be compromised.

IV. DEFENSE MECHANISM

A. OVERALL SCHEME AND ASSUMPTIONS

1) ASSUMPTIONS AND DEFINITIONS OF NOTATIONS

The mechanism relies on the following assumptions: 1) The terminals adopt the WLAN-based positioning, and the server has previously stored the geographic location information of the known BSs; 2) The density of users near the BS should be large enough, i.e., there are at least two users near each BS;

TABLE 1. Threats of the attacks on selected android applications.

Application name	Threats					
	wrong positioning	error message pushing	social network information leaking			
			nickname	identity	photos	relationship
aMap[1]	✓	✓	-	-	-	-
Twitter[2]	✓	-	✓	✓	✓	✓
Sina Weibo	✓	-	✓	✓	✓	✓
WeChat[3]	✓	✓	✓	✓	✓	-
QQ[4]	✓	✓	-	-	-	-
QQ browser[5]	✓	-	-	-	-	-
Alipay[6]	✓	✓	✓	✓	-	-

[1] http://lbs.amap.com/, an application for traffic navigation
[2] https://www.twitter.com/, an application for social networking
[3] http://www.wechat.com/en/, an application for social networking
[4] https://im.qq.com/index.shtml, an application for social networking
[5] http://browser.qq.com/, an application for mobile payment
[6] https://www.alipay.com/, an application for traffic navigation

TABLE 2. The definitions of notations.

Notation	Description
Z	the set of information collected from BSs
w	the BS tag
N	the number of BSs scanned by a device
n	the number of devices
x	the random value of strong extractor's (Ext) input
P	the auxiliary data generated in Generation (Gen) process
s	the generated string of Security Sketch (SS) process
r	the random value of SS′s input
t	the time of terminal collecting characteristic parameters or requesting services
T	the time period
R/R'	the key value generated/reconstructed by FE

3) The attacker can only weaken the local BSs' signals and can't shield them all; 4) The server and users are credible, which means the attacker can only launch external attacks.

The descriptions of notations used in this paper are showed in the TABLE 2.

2) THE DEFENSE SCHEME

The main idea of our proposed mechanism is to verify the validity of BSs using the reproducibility and unpredictability of BS tags with the time and location attributes, which can prove the spatial-temporal authenticity of location and ensure the robust localization. The reproducibility means that two measurements at the same space and time yield two BS tags according to the same BS, and the two tags could match with high probability. The unpredictability is equivalent to unforgeability. That is, an adversary, not at the specific place and time, is unable to produce a BS tag that matches the tag constructed at that location and time.

The defense mechanism includes the server (S), the witness (W) and the user (U), as shown in Fig. 5. All Ws and Us are mobile users. We divide the cycle T according to the

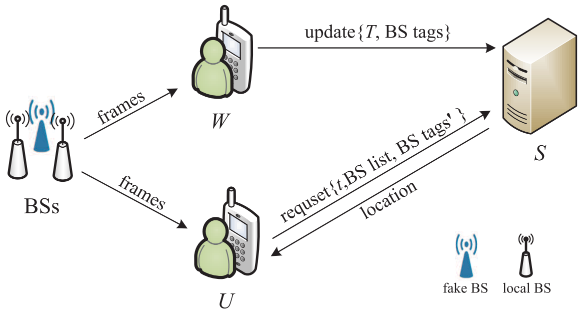


FIGURE 5. The overall architecture of defense mechanism. Its components include the server (*S*), the witness (*W*) and the user (*U*) except BSs. All *W*s and *U*s are mobile users.

source BS time so that no time synchronization is required. In each cycle T, each W constructs a corresponding BS tag for neighboring BSs by using bloom filters to fuse corresponding 802.11 frames. After collecting BS tags from all Ws, S select a BS tag as the datum tag (DTag) for each BS by applying FE to verify the legality and the consistency of both BS tags and BS locations. Thus, each BS only has a DTag stored in S. When a U requests location-based services, he reports the BS list and the stored last-cycle BS tags to S. S verifies the legal BSs from the $U's$ BS list by the consistency verification of BS tags and DTags. And then, S calculates the positioning result and returns it to U.

a: UPDATING DATUM TAGS

Step 1: Ws scan the frames of the nearby BSs to construct a BS tag for each BS in each T and then upload the message $(U_1^{n_k} W_j, U_1^{N_k}(BS_i, tag_i, key_i, P_i), t_k)$ to S, as the detailed protocol design illustrated in Fig. 6, where n_k is the number of Ws who upload BS tags and $(U_1^{n_k} W_j$ is all information of Ws, t_k is the current period, N_k is the number of BSs scanned by W_j in t_k, tag_i is the BS tag corresponding to BS_i generated by W_j, key_i and the P_i are two auxiliary data to verify BS tags. The following part B describe the generation of BS tags. In addition, each W stores BS tags of the latest cycle in case of on-demanding request services.

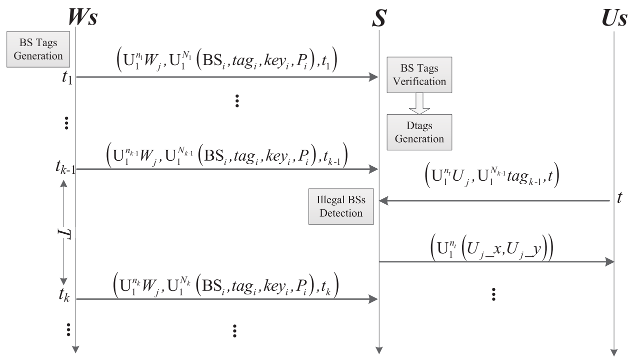

Ws
S
Us
BS Tags Generation
$\left(U_1^{n_1} W_j, U_1^{N_1}(BS_i, tag_i, key_i, P_i), t_1\right)$
t_1
BS Tags Verification
Dtags Generation
$\left(U_1^{n_{k-1}} W_j, U_1^{N_{k-1}}(BS_i, tag_i, key_i, P_i), t_{k-1}\right)$
t_{k-1}
$\left(U_1^{n_t} U_j, U_1^{N_{k-1}} tag_{k-1}, t\right)$
t
Illegal BSs Detection
T
$\left(U_1^{n_t}(U_{j}\_x, U_{j}\_y)\right)$
$\left(U_1^{n_k} W_j, U_1^{N_k}(BS_i, tag_i, key_i, P_i), t_k\right)$
t_k

FIGURE 6. The protocol of the defense mechanism. *W*s periodically upload messages as evidences and *U*s request service on demand.

Step 2: S filters out the legitimate BS tags to be DTags and updates database, which contains the known legal BSs information, including their IDs, geographical locations, and the corresponding DTags of the latest period T. The generation of DTags is depended on two factors: BS tags and BS locations. Only Ws that are really close to the same BS can construct highly similar BS tags. BSs scanned by a W should be in roughly the same region. Only the results of verifications of BS tags and BS locations are both consistent, it can be DTag. As it can be seen in the part C of this section.

b: VERIFYING BS

Step 1: The scanned BS list at some time t maybe different from the previous cycle, so the Us only report the intersection BS tags of last cycle to initiate the request. Us send the request message $(U_1^{n_t} U_j, U_1^{N_{k-1}} tag_{k-1}, t)$to S, where t is the current requesting time, n_t is the number of Us who initiate the request, $(U_1^{n_t} U_j$ is all information of Us, N_{k-1} is the number of intersection BS tags stored by Uj in the previous cycle t_{k-1} and $U_1^{N_{k-1}} tag_{k-1}$ is BS tags generated by U_j.

Step 2: S compares the BS tags with DTags to select the legal BSs near the Us. Once legal BSs are determined, S could implement the positioning algorithm and then send the positioning result $(U_1^{n_t}(U_{j-}x, U_{j-}y))$ as the message to Us, where $U_{j-}x$ and $U_{j-}y$ are the x coordinate and y coordinate of the location respectively.

B. BS TAGS GENERATION

The W scans the surrounding BSs' signals, extracts some parts related to location as characteristic parameters (*i.e*, keys), and maps then into a BS tag using a bloom filter. We use entropy to describe the BS tag's unforgeability. The larger the entropy, the greater the uncertainty of kays. Therefore, more information is needed to determine the parameters, which lowers the success rate to predict BS tags. In other words, the higher the unpredictability, the stronger resistance against to the location spoofing attack.

1) CHARACTERISTIC PARAMETER EXTRACTING

In order to guarantee the unforgeablity of BS tag, the frames should satisfy high entropy, which is depending on the type of 802.11 frames. Meanwhile, to reducing the computing overhead of hash mappings, the quantity of BS frames cannot be too large. Motivated by this, beacon frames and probe frames are chose to generate BS tag in our mechanism. Supposed the BS frames collected by the user's device is Z, and the information extracted to construct tag is X. Then, we have,

$$X_i = Fil(f_{BS}(Z, t_i)) \tag{1}$$

where t_i denotes the time when W collecting frames, $f_{BS}(.)$ denotes the function to filter the proper frames from Z to construct the set Y, $Fil(.)$ denotes the function that chooses MAC header from Y as the element of X_i. The terminal adds the effect parts into the X set when scanning a characteristic frame.

The MAC header of 802.11 management frame is composed of frame control, duration/ID, MAC address and sequence control. frame control and duration/ID are similar for the same type of frame. For example, those of the Beacon frame are 0×80 and 0×00 respectively, and those of the Probe response frame are 0×50 and 0×31 respectively. For different types of 802.11 frames and different environments, the numbers of source and destination of the MAC address domain are different. sequence control is changing over time. When a frame is received, the sequence number is incremented by 1.

We treat the MAC address of 802.11 frames as a characteristic parameter, which can reflect the interactive relationship between the BS and surrounding environment.

Capturing BS frames is a discrete event related to the time sequence, which is considered as a Markov process. According to the definition of Shannon entropy, the higher the entropy, the harder it is to forge frames. The entropy of Y is calculated as in

$$H(Y) = E(I(y)) = \sum_{n=1}^{m} p_i I(y_i) = -\sum_{n=1}^{m} p_i \log p_i \tag{2}$$

where p_i is the probability that the $i-th$ frame is y_i given $(i-1)-th$ frame y_{i-1}, calculated by $p(y_i \mid y_{i-1})$. More precisely, p_i is calculated by the ratio of the amount of frame (y_{i-1}, y_i) occurred sequentially to all transit cases, as in Equation (3).

$$p_i = \frac{C(y_i, y_{i-1})}{\sum_{i-1}^{m} C(y_m, \ldots, y_{i-1})} \tag{3}$$

2) HASH MAPPING

We use the hash mapping to compress parameters with different lengths into a fixed length of data to reduce the storage overhead of the system. The irreversibility and randomness properties of the hash mapping reduce the likelihood that the output of X is inferred by the attacker. Ws construct BS tags corresponding to the scanned BSs.

As a single hash is easy to cause the collision problem, a bloom filter with m hash functions is used to dispose of X. A bloom filter firstly constructs an n-bit binary string w of all zero, and then uses $Ins(h_i(X_j), w)$ as its insert procedure, where $1 \leq i \leq m$ and $1 \leq j \leq num(X)$. More precisely, $num(X)$ denotes the number of X and the function $Ins()$ adds an element, feeding it to each of the m hash functions to get m array positions. Function h_i sets the bits at all these positions to 1. That is, X_j is put the position $h_i(X_j)$ to be 1. Finally, an n-bit string w constructed with the input X is a BS tag. Taking three hash functions for an example, the structure that inserts location characteristic X into a binary string w is illustrated in Fig. 7.

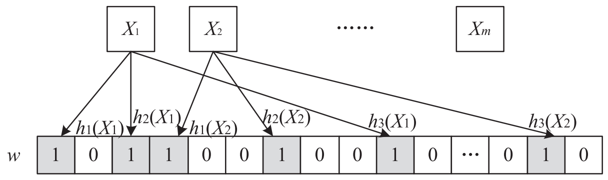


FIGURE 7. Characteristic parameters mapping to a binary string, taking 3 hash functions for an example.

In the initialization phase, the server sets the structure of the bloom filter with m hash functions. Then, each terminal uses the same bloom filter to dispose of the set X. The W iterative maps the BS frames into a binary string w with len bits, which we called the BS tag. The pseudo code of generating BS tags is presented in Algorithm 1. Taking security into consideration, the hash functions should be updated periodically by S, and Ws renew their BS tags with the new bloom filter.

Algorithm 1 The Generation of BS Tags

```
Input: $X$, $m$ hash functions, $N$(the number of scanned BSs by a $W$)
Output: BS tags
1:  for $i = 1$ to $N$ do
2:      construct a binary string $w_i$ for each BS;
3:      for $j = 1$ to $m$ do
4:          for $k = 1$ to $num(X)$ do
5:              $w_i \leftarrow Ins(h_j(X_k), w_i)$;
6:          end for
7:      end for
8:      add $w_i$ to BS tags;
9:  end for
10: return BS tags;
```

C. BS TAGS VERIFICATION

The BS tags constructed by different users at the same time interval and at one place corresponding to the same BS may not exactly consistent due to the high-bit error rates in wireless transmission. The fault-tolerant FE are used to verify the consistency of BS tags.

Generally, The FE is constructed by a Generation process (Gen) and a Reproduction process (Rep). They can extract a uniform and random string as the key R from a lot of biometric data, while tolerating certain noise. Most fuzzy extractors are constructed based on error-correcting codes. Thus, a fuzzy extractor allows one to extract some randomness r from w and then successfully reproduces r from any w_1 that is close to w. In our scheme, we use FE to verify the consistency of BS tags. If FE can refactor the same key in a certain fault-tolerant range, the two BS tags are consistent, which is represented as in Equation (4).

$$\begin{cases} Gen(w) = (P, R) \\ Rep(w', P) = R' \end{cases} \tag{4}$$

In the Gen process, after inputting the BS tag w, FE can extract the key value R and auxiliary data P, where P won't reveal many secrets of R. In the Rep process, using w' given the P, FE can reconstruct the R' if $w' \sim w$, and then $R' = R$.

FE are constructed by Ext and SS, as shown in Fig. 8. The randomness x and the output of SS s can be stored as a helper string P. Secure sketch makes it possible to reconstruct noisy input, so if the error rate of the new input is within a certain threshold, it is possible to reconstruct the original input.

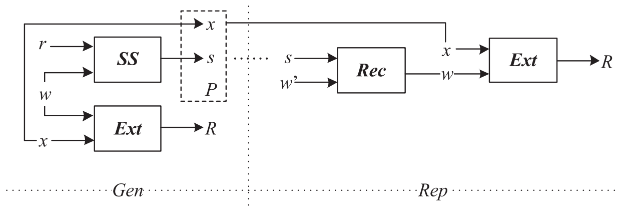


FIGURE 8. FE with security sketch.

In SS process, the inputs are the BS tag w and the sketch s. In the Recover (Rec) process, given s and w' that is close to w, it is possible to reconstruct w. A random value r needs to be injected into the SS runtime to prevent the attacker from recovering w. The model of security sketch is represented as in (5).

$$\begin{cases} SS(w, r) = s \\ Rec(w', s) = w \end{cases} \tag{5}$$

In hamming space, the computational process of a security sketch is showed in Equation (6). In the SS process, the random code r is calculated by the $Enc(r)$ processing with error correction coding. The original input w and $Enc(r)$ results of an exclusive or operation are taken as the output s of $SS(w, r)$. The Rec process decodes s and w'. If $w' \sim w$, namely $dis(w, w') \leq \Phi$, where $dis()$ is the function to calculate the statistical distance and Φ is ECC fault-tolerant bits, the Rec process can reconstruct the initial input w. The statistical distance can be calculated by the hamming distance, i.e., $dis(w, w') = Ham(w, w')$ as the BS tag w is a binary string.

$$\begin{cases} SS(w, r) = w \oplus Enc(r) = s \\ Rec(w', s) = Dec(w' \oplus SS(w, r)) = w \end{cases} \tag{6}$$

If $dis(w, w') < \Phi$, the reproduced w' and the original input w are very similar or even the same. Then the key values

```
Algorithm 2 The Verification of BS Tags
Input: BS tags set TAG, N
Output: classified BS tags STAG
 1: $STAG \leftarrow \emptyset$;
 2: while TAG $\neq \emptyset$ do
 3:     Fetch $w$ from TAG;
 4:     $W \leftarrow \{w\}$
 5:     for each tag $w'$ in TAG do
 6:         for $i = 1$ to $N$ do
 7:             $(R_i, P_i) \leftarrow Gen(w_i)$;
                $R'_i \leftarrow Rep(w'_i, P_i)$;
 8:             if $R = R'$ then
 9:                 $W \leftarrow W \cup \{w'\}$;
                    $TAG \leftarrow TAG - \{w'\}$;
10:             end if
11:         end for
12:     end for
13:     $CTAG \leftarrow CTAG \cup W$;
14: end while
15: return CTAG;
```

R extracted in the two processes are matched, i.e., the two BS tags are consistent. The pseudo code of verifying the consistency of BS tags is showed in Algorithm 2. For each BS, the BS tag constructed by users may be classified into different types.

D. DTags GENERATION

In this work, we use a double verification of BS tags and BS locations to perform filtering of legal BS and Dtags. S only needs to store a DTag for each BS, which can guarantee the timeliness while reducing the memory space occupied. If both the BS tags and BS locations are consistent, then the BS is legal and can determine the corresponding DTag. The BS is considered to have changed its location if only one of the verification results is consistent. If neither BS tags nor BS locations is consistent, the BS is regarded to be disguised.

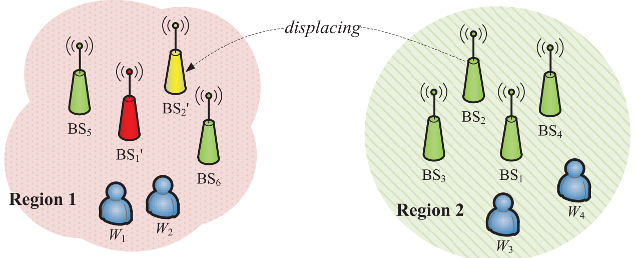


FIGURE 9. The sample graph of selecting the legal BSs.

As shown in Fig. 9, there are four users $W_1 \sim W_4$ who periodically scan nearby BS broadcast frames. In region 1, users can capture the frames from BS1′, BS2′, BS5 and BS6. Among these BSs, BS1′ is disguised as legal BS1 in region 2 and BS2′ is moved from region 2 which location information hasn't been updated. In region 2, users can capture the frames from BS1, BS3, and BS4. For BS2′, its location is quite different from other BSs in region 1, so the verification of BS locations is inconsistent. At the same time, W_1 and W_2 are in the vicinity of BS2′, so the verification of BS tags is consistent. Hence the position of BS2 is changed. For BS1′, both W_1 and W_3 can receive the frames broadcasted from the same ID, while the two users are in different places, so the BS tags generated by W_1 and W_1 are inconsistent and positional deviation is large, suggesting that BS1′ is fake.

Theoretically, the locations scanned by different users based on the same BS should be in a roughly same area. The consistency of BS locations is defined as the BS distance deviation being falling within a certain range. The change of BS position happens rarely. After verifying the BS tags, S verifies the consistency of BS locations for further operation. The pseudo code of generating DTags is showed in Algorithm 3.

```
Algorithm 3 The Generation of DTags
Input: classified BS tags STAG
Output: DTags
 1: for each type of BS tags in STAG do
 2:     calculate the center point $(addx, addy)$of the consistent
        BS tags;
 3:     for each BS in this type do
 4:         $d \leftarrow sqrt((x - addx)^2 + (y - addy)^2)$
 5:         if $d < \triangle$ then
                // $\triangle$ is the threshold
 6:             BS locations are consistent, continue;
 7:         else
 8:             BS locations are inconsistent, its location displac-
                ing, break;
 9:         end if
10:     end for
11:     the BS tag is a DTag;
12: end for
13: return DTags;
```

E. ILLEGAL BSs DETECTION

S compares BS tags with DTags to detect the illegal BSs near U. The pseudo code of detecting the illegal BSs is shown in Algorithm 4. Similar to Algorithm 2, S selects the valid BSs by using FE, while the same type BS tags in Algorithm 2 means that the BS is legal here. The consistency of BS tags and BS locations have been verified, so verifying the reported BS tags is sufficient here. After selecting the legal BSs, S uses them to calculate the position of U and responses to the request from Us'.

V. EXPERIMENTAL ANALYSIS

A. SOURCES OF BS FRAMES

In order to satisfy the unpredictability requirement, a good BS tag should be time-variant and has high entropy. There are various types of frames in WLAN. We measured the entropies

Algorithm 4 The Detection of Illegal BSs

```
Input: the BS tags and DTags
Output: legal BSs
1:  for each $U$ and BS do
2:      $w \leftarrow$DTag, $w' \leftarrow$BS tag reported by $U$;
3:      $(R, P) \leftarrow Gen(w)$;
4:      $R' \leftarrow Rep(w', P)$;
5:      if $R! = R'$ then
6:          the BS is illegal;
7:      else
8:          add BS to legal BSs set of the $U$;
9:      end if
10: end for
11: return legal BSs;
```

of some main 802.11 MAC headers to analyze how to select the features.

The statistical data of each frame type is shown in Fig. 10, which is captured at the same teaching place corresponding to the specified BSs at three experimental times by AirMagnet WiFi Analyzer PRO [24]. The three frame collections which are captured in the same time interval are as follows. (a) The five types of frames, i.e., Beacon, Probe request, Probe response, RTS, CTS and ACK are captured for 55.66s on a Wednesday morning. (b) On a Saturday morning, the five types of frames collected are Beacon, Probe request, Probe response, Authentication and Data during the same time period. (c) On a Saturday evening, the five types of frames collected are Beacon, Probe request, Probe response, RTS and ACK. These experiments collected more than 3,000 frames, which are detailed in TABLE 3. As shown in Fig. 10, Beacon frame accounts for a large proportion of all types of BS frames because it broadcasted regularly to indicate the existence of the BS. Probe request frame is initiated by the terminal to detect nearby BSs. We can see that the Probe request frame accounts for 2% in Fig. 10(a) and (b), while almost no in Fig. 10(c). The data collection time of (a) and (b) is in class time, so there are many active users around the specified BS.

Although the ratios of different frames are quite different at the three different times in the experiment, their broadcasting rules are quite similar. Their entropies are calculated using Equation (2) and are shown in Fig. 11. Beacon frames broadcast periodically with continuous SN (Sequence Number) at a 0.1024s time interval. Its broadcasting rule is the most obvious, while the entropy is relatively small. The RTS/CTS frame safeguards the non-conflict communications between terminal and BS, so the entropy is depended on the communication requirements of the access devices. The time interval and frame format of Probe frame depend on the device driver and terminal access pattern [25]. Probe frame has close relationship with BS's surrounding equipment, so it has the highest entropy. The effect part of domain address which has high entropy can be considered as important characteristic parameters, which not only reflect the interactive relationship between BS and the surrounding environment, but also help reduce the storage overhead.

Taking the data RTS and CTS in TABLE 3(a) for an example, the entropy is calculated respectively 5.89 bit and 6.66 bit when 2 is as the base for the *log* arithmic function. As a result, the probability of an attacker predicting these

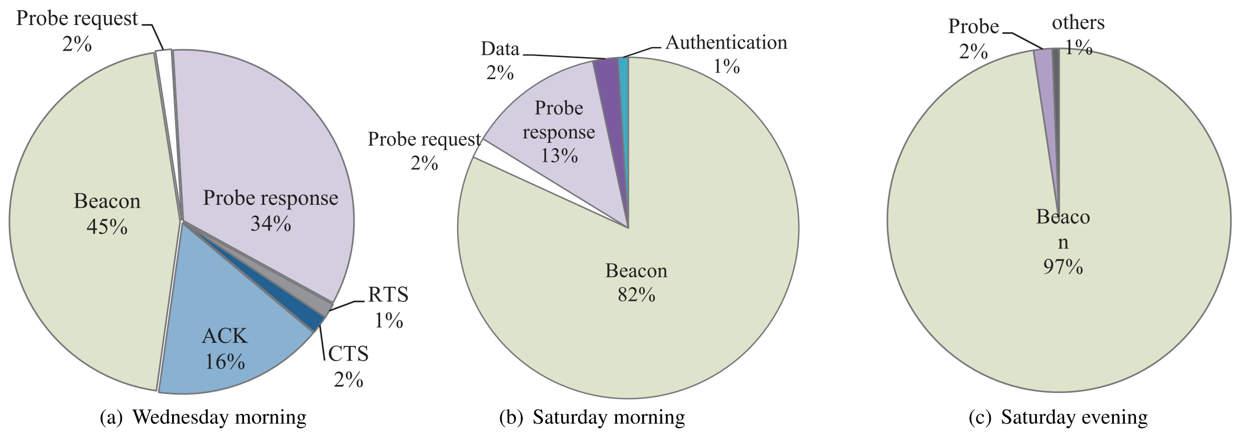


FIGURE 10. Traffic Summary of three data collections. From the figures, we find that Beacon frame accounts for a large proportion amongst all types of BS frames.

TABLE 3. Packet counts.

frame / number	Beacon	Probe request	Probe response	RTS	CTS	ACK	Data	Authentication
(a)	1535	51	1149	47	58	547	-	-
(b)	2544	60	404	-	-	-	74	30
(c)	3064	2	53	2	-	18	-	-

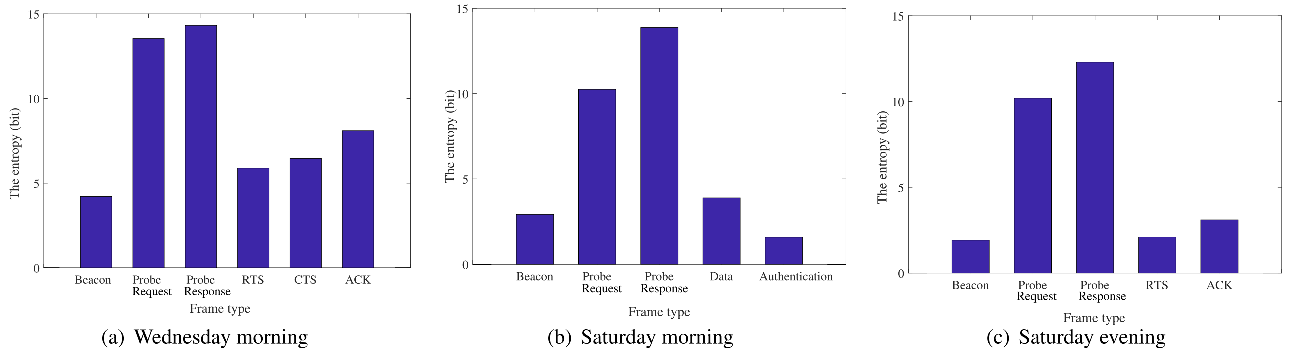


FIGURE 11. **The 802.11 frame headers entropies at the three times. The entropies of Probe frames are larger than those of other type frames.**

two types of frames is approximately 0.0168 and 0.0099, respectively. Based on the above analysis, we select the MAC header of Probe request and Probe response frame with high entropies to be the element of set X. Suppose that the entropy of a characteristic parameter is a bit, the probability of X being correctly predicted is 2^{-a}. Therefore, there is a low probability for an attacker to forge a complete and correct BS tag.

B. COMPUTATIONAL OVERHEAD

In the process of generating BS tags, each user constructs a BS tag for each neighboring BS. The number of hash mappings used in the bloom filter is considered constant. Therefore, the time complexity of generating BS tags is $O(n^2)$, which includes the two processes of characteristic parameter extraction and hash mapping. Apparently, the latter depends on the $num(X)$ and hash functions used. Taking the efficiency of the system into account, the most intuitive approach is to reduce the digits of w which is generated by the bloom filter. But the smaller the length of w is, the less likely it is to ensure the system reliability due to the existence of false positive rate of bloom filter. We use Lenovo lemon K3 with 1.5GHz quad-core Qualcomm snapdragon 410 (MSM8916) CPU and 1G memory as the terminal used the hash functions based on SHA-512 algorithm, and its running time is about 3.5ns. The time complexity of the verification of BS tags is $O(n^2)$. The FE adopt BCH coding in the algorithm to verify the consistency of BS tags since w is a binary string in Hamming distance space. The coding and decoding algorithms of error correcting code in the *Gen* and *Rep* processes occupy most of the execution time. We use a 32-bit PC with 2.70GHz Intel(R) Pentium(R) G630 CPU as the server to verify, the BS tags based on the receipt of the requests from Us. Each verification process requires one decoding and one encoding, which takes about 13μs, as shown in Fig. 12(a).

The length of the BS tag, the number of hash functions, and the number of X parameter sets (i.e., *len*, m and $num(X)$) all affect the actual calculation. The false positive rate involves these three parameters. Therefore, we discuss the false positive rate here. The false positive rate of the bloom filter is discussed as follows. The probability is $1 - 1/len$ that one digit isn't set to 1 after inserting a X_i by a hash function. The probability of the digit isn't set to 1 is $(1 - 1/len)^{m \cdot num(X)}$ after inserting all elements of X by m hash functions. Then the probability of the digit is set to 1 is $1 - (1 - 1/len)^{m \cdot num(X)}$. So the false positive rate of bloom filter can be seen as the following function. When $m = ln2 \cdot len/num(X)$, the false positive rate is minimum.

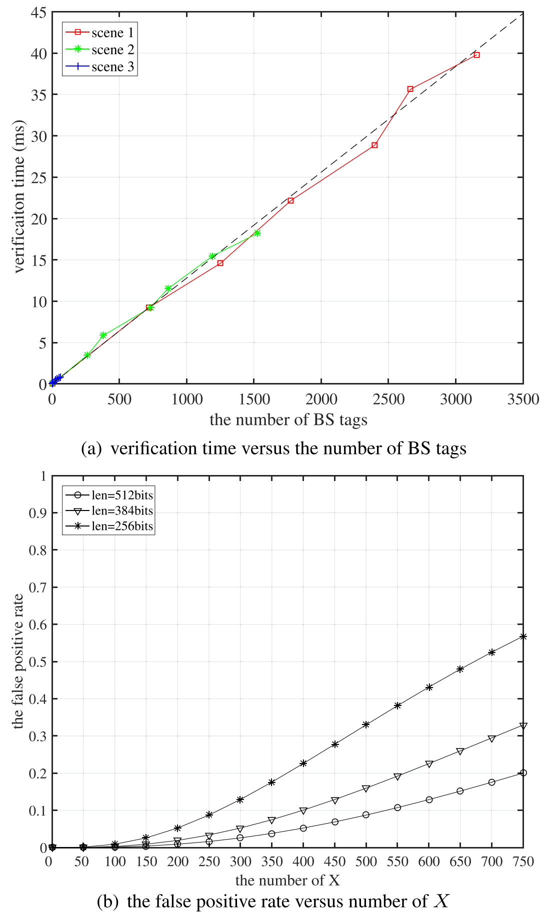


FIGURE 12. **The analyses of the factors affecting the computational overhead. In (a), the verification time increases with the increase of the number of BS tags. In (b), with the growth of *num*(*X*) and the decrease of len, the false positive rate increases gradually.**

$$P(m) = (1 - (1 - 1/len)^{m*num(X)})^m \sim (1 - e^{-m*num(X)/len})^m \tag{7}$$

It's obviously that $m \geq 1$, transforming into $len/num(X) \geq 1/ln2$, which means that the size of w almost linear synchronously changes with the size of set X. As shown in Fig. 12(b), with the growth of $num(X)$ and the decrease of len, the false positive rate increases gradually. So it should keep the quantity of X proper and use the appropriate hash functions which determine the length of BS tag when extracting the characteristic parameters.

C. STORAGE AND COMMUNICATION OVERHEAD

As can be seen in TABLE 3(a), the number of Probe response frames is 1149, and we store their destination address fields (using 6 bytes) as the BS tag. If a hash table is used to store the data, it needs (1149*6bytes)/50%≈13.5KB in the case that the hash table is half-full to ensure that the storage efficiency is not more than 50%. When taking a bloom filter with 6 hash functions, it occupies (1149*6bits)/50%≈1.7KB under the circumstance that the space utilization rate is 50%. Storing the same data, the space consumed for using bloom filter to store the BS tag is roughly about 1/8 of that of the hash table.

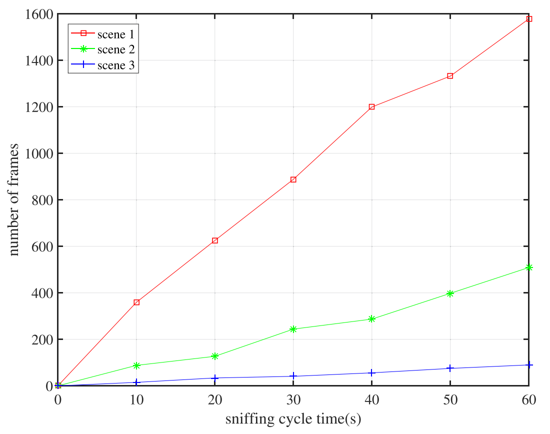
Our experiments are implemented on a working day and during the weekend in the same teaching area. Because the working day falls in the teaching semester, there are many students in the area. As the students carry a large number of smartphones, the number of Probe frames is therefore higher than that of the weekend evening. Obviously, as the sniffing cycle time increases, the number of collected characteristic frames increases, as shown in Fig. 13(a). BS tags are made up of characteristic frames, which are compressed into w at each in Fig. 13(b). If the number of hash functions is fixed, there is a nearly linear relationship between the number of frames and the size of BS tags. In this case, the memory of BS tags is less than 6KB, which is well manageable by the smartphones. The server stores only a Dtag for a single BS. Also, if the terminal sends the BS tags to the server for storage, the communication overhead is only a few bytes.

D. ANALYSIS OF T

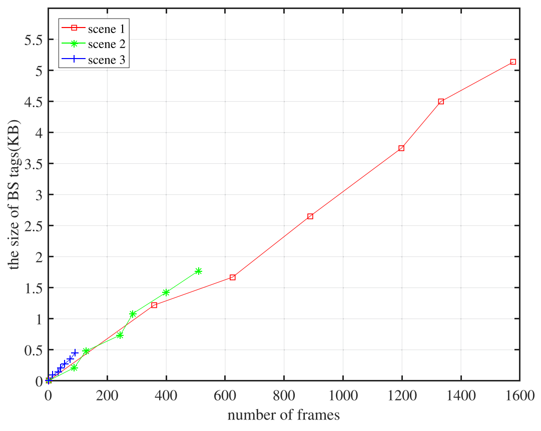
The cycle T is determined by W's moving speed (v) and the system overhead. We assume that the moving distance is less than 2 times of the BS broadcast radius. Under the assumption that Ws and BSs are evenly distributed in a target field, there is a linear relationship between the overlapping area of the two times and the number of overlapping BSs. The larger the overlapping area it is, the more the BSs, the more the same BSs scanned in the two cycles. The random deployment of the BSs with a density p_b can be modeled as a spatial homogeneous Poisson point process of rate p_b. Thus, the probability that there are at least k BSs deployed within an area of size s, is given by the Poisson distribution:

$$p = p(X > k) = 1 - \sum_{i=0}^{k} \frac{\lambda^i}{i!} e^{\lambda} \tag{8}$$

$$\begin{aligned}\lambda &= p_b * s \\ &= p_b * [r^2 arccos(\frac{d/2}{r}) - d\sqrt{r^2 - (d/2)^2}] * 2\end{aligned} \tag{9}$$



(a) number of frames versus sniffing cycle time



(b) the size of BS tags versus number of frames

FIGURE 13. Storage and computing overhead. Obviously in (a) as the sniffing cycle time increases, the number of collected characteristic frames increases. And in (b), as the sniffing cycle time increases, the size of the BS tags corresponding increases gradually.

where λ is the average incidence of scanning the overlapped BSs from the unit area which can be calculated by Equation (9), s is the overlapped areas of $W's$ two cycle scanning, r_0 is the radio radius of BS and d is the moving distance in the cycle, that is, $d = v{*}T$. As shown in Fig. 14, we find that p decreases with the increase of T and v. A smaller the T will lead to a slower v, the more overlapped BSs and a greater p, but the communication cost and calculation of the system will also increase at the same time. In order to facilitate the verification of BS tags and positioning, the number of BSs in the overlapping area should be at least 5 (to ensure the robustness of positioning). So the value of T should be set as a relatively large value which can also reduce the computational overhead. For example, when the value of p is appropriate (i.e., $p > 0.9$) and $v = 1.5$m/s, T can be reduced to 120s.

VI. SECURITY ANALYSIS

In this part, we attempt to show our proposal provides unforgeable (unpredictable) location tags and satisfies security. Formally, we have:

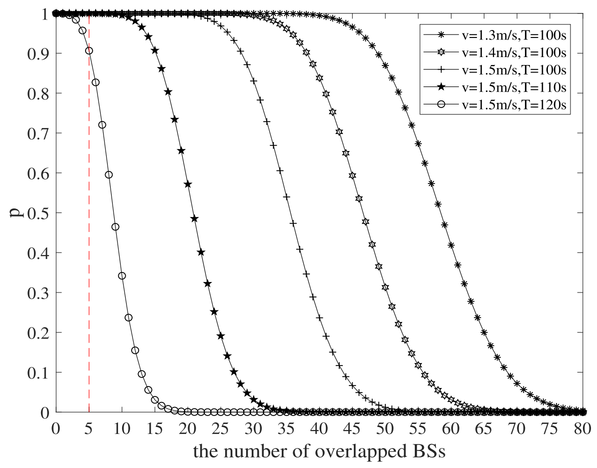


FIGURE 14. The relationship between moving speed, cycle and overlapped BSs, that is, p decreases with the increase of T and v.

Lemma 1: The entropy of frames constituting the BS tag is high enough to ensure that the probability of forging the BS tag is low enough for security. i.e., the BS tag is unforgeable.

Prove: The probability of forging the BS tag is $p(x = w) = 2^{-a}$, where x is the BS tag forged by the attacker, w is the original BS tag and a is the entropy of BS frames. A BS tag is unforgeable if $p(x = w) = \epsilon$. Small ϵ means bigger bit and greater difficulty of forging the BS tag. For example, when $a > 10$ (bit), the probability of forging a BS tag is less than 0.0009 which approaching 0. On the other hand, the BS tag is generated by using a one-way irreversible bloom filter. Even if the attacker captures a real BS tag w, the corresponding characteristic parameters X and the hash functions used by the bloom filter cannot be analyzed. Therefore, the BS tags in our mechanism are unforgeable.

Lemma 2: If P is constructed by the output s of the SS process and random seed x, P is safe.

Prove: The auxiliary data P generated in *Gen* process is public, formed by s and x. Random value r enhances the entropy of s, ensuring different output even under the same input w. In Equation (5), if there is no input of r, then w is calculated through a fixed function to get s in the SS process. So the random r input is necessary to ensure that P won't reveal too much information about R and it's useless to an attacker.

Lemma 3: If *Ext* is implemented by HMAC and x is random, the key value R is safe.

Prove: The algorithm of HMAC uses a message authentication code to ensure the integrity of information, so an attacker is unable to know R in advance. HMAC creates a message involving a cryptographic hash function and a secret cryptographic key. The random seed x is agreed in advance and updated irregularly. The strong extractor is defined as *Ext*: $\{0, 1\}^{|w|} \times \{0, 1\}^{|x|} \rightarrow \{0, 1\}^{|R|}$, where $|w|$, $|x|$ and $|R|$ are the length of w, x and R, so the R generated by the FE is close to the uniform distribution [26], [27]. Unless the attacker has the same P and similar w, it's really difficult to obtain the accurate value of R, even when the attacker colludes with some user to deceive the server.

VII. CONCLUSION

Wireless positioning systems are susceptible to spoofing attacks, which further pose a more serious threat to privacy of MSNS than ever before. We first show a privacy attack model base on location spoofing attack in MSNS, and then propose a defense mechanism based on BS tags for authenticating the spatial-temporal property of a geolocation, while preventing the spoofing attacks on WLAN based positioning. The proposal verifies the legitimacy of BS based on the unpredictability and reproducibility properties of BS tags. Furthermore, we utilize the bloom filters and fuzzy extractors to reduce storage overhead and tolerate channel errors respectively. The future research includes reducing storage and communication overhead and looking for frames with higher entropy.

REFERENCES

[1] X. Ni, J. Luo, B. Zhang, J. Teng, X. Bai, B. Liu, and D. Xuan, "A mobile phone-based physical-social location proof system for mobile social network service," *Secur. Commun. Netw.*, vol. 9, no. 13, pp. 1890–1904, 2016.

[2] *Sina Weibo*. Accessed: Dec. 15, 2018. [Online]. Available: https://weibo.com/

[3] A.-Y. Zhou, B. Yang, C.-Q. Jin, and Q. Ma, "Location-based services: Architecture and progress," *Chin. J. Comput.*, vol. 34, no. 7, pp. 1155–1171, Sep. 2011.

[4] N. O. Tippenhauer, K. B. Rasmussen, C. Pöpper, and S. Čapkun, "Attacks on public WLAN-based positioning systems," in *Proc. 7th Int. Conf. Mobile Syst., Appl., Services (Mobisys)*, 2009, pp. 29–40.

[5] K. Jindal, S. Dalal, and K. K. Sharma, "Analyzing spoofing attacks in wireless networks," in *Proc. 4th Int. Conf. Adv. Comput. Commun. Technol.*, Feb. 2014, pp. 398–402.

[6] L. Qing, Y. Ayong, and X. Li, "Research on privacy attack based on location cheating in social network," *Netinfo Secur.*, vol. 2017, no. 5, pp. 51–56, 2017.

[7] B. Henne, C. Szongott, and M. Smith, "SnapMe if you can: Privacy threats of other peoples' geo-tagged media and what we can do about it," in *Proc. 6th ACM Conf. Secur. Privacy Wireless Mobile Netw. (WiSec)*, 2013, pp. 95–106.

[8] J. Cho, J. Yu, S. Oh, J. Ryoo, J. Song, and H. Kim, "Wrong siren! A location spoofing attack on indoor positioning systems: The starbucks case study," *IEEE Commun. Mag.*, vol. 55, no. 3, pp. 132–137, Mar. 2017.

[9] M. Cunche, "I know your MAC address: Targeted tracking of individual using Wi-Fi," *J. Comput. Virol. Hacking Techn.*, vol. 10, no. 4, pp. 219–227, Dec. 2013.

[10] C. Matte, J. P. Achara, and M. Cunche, "Device-to-identity linking attack using targeted Wi-Fi geolocation spoofing," in *Proc. 8th ACM Conf. Secur. Privacy Wireless Mobile Netw. (WiSec)*, 2015, pp. 1–6.

[11] P. Zhang, S. G. Nagarajan, and I. Nevat, "Secure location of things (SLOT): Mitigating localization spoofing attacks in the Internet of Things," *IEEE Internet Things J.*, vol. 4, no. 6, pp. 2199–2206, Dec. 2017.

[12] D. Qiu, B. Dan, and S. Lo, "Robust location tag generation from noisy location data for security applications," in *Proc. Int. Tech. Meeting Inst. Navigat.*, 2009, pp. 586–597.

[13] Z. Lin, D. F. Kune, and N. Hopper, "Efficient private proximity testing with GSM location sketches," in *Financial Cryptography and Data Security*. Berlin, Germany: Springer, 2012, pp. 73–88.

[14] Y. Zheng, M. Li, W. Lou, and Y. T. Hou, "Location based handshake and private proximity test with location tags," *IEEE Trans. Dependable Secure Comput.*, vol. 14, no. 4, pp. 406–419, Jul. 2017.

[15] Y. Zheng, M. Li, and W. Lou, "SHARP: Private proximity test and secure handshake with cheat-proof location tags," in *Computer Security—ESORICS*. Berlin, Germany: Springer, 2012, pp. 361–378.

[16] S. A. Akinboro, A. Omotosho, and M. Odusami, "An improved model for securing ambient home network against spoofing attack," *Int. J. Comput. Netw. Inf. Secur.*, vol. 10, no. 2, pp. 20–26, Feb. 2018.

[17] J. Yu, E. Kim, H. Kim, and J. Huh, "A framework for detecting MAC and IP spoofing attacks with network characteristics," in *Proc. Int. Conf. Softw. Secur. Assurance (ICSSA)*, Aug. 2016, pp. 49–53.
[18] P. Bahl and V. N. Padmanabhan, "RADAR: An in-building RF-based user location and tracking system," in *Proc. 19th Joint Conf. IEEE Comput. Commun. Societies*, Mar. 2000, pp. 775–784.
[19] *WiGLE: Wireless Network Mapping. Wireless Geographic Logging Engine*. Accessed: Dec. 2, 2017. [Online]. Available: https://wigle.net/
[20] Universal Software Radio Peripheral (USRP). *Ettus*. Accessed: Mar. 21, 2019. [Online]. Available: http://www.ettus.com
[21] MDK3 Package Description. *ASPj of K2WRLZ*. Accessed: Aug. 15, 2019. [Online]. Available: http://tools.kali.org/wireless-attacks/mdk3
[22] P. Sepehrdad, P. Susil, and S. Vaudenay, "Smashing WEP in a passive attack," in *Proc. Int. Workshop Fast Softw. Encryption*. Berlin, Germany: Springer, 2013, pp. 155–178.
[23] *Location Service-Weibo API*. Accessed: Aug. 17, 2018. [Online]. Available: http://open.weibo.com/wiki/%E4%BD%8D%E7%BD%AE%E6%9C%8D%E5%8A%A1
[24] Fluke. *AirMagnet WiFi Analyzer PRO|NETSCOUT*. Accessed: Mar. 22, 2017. [Online]. Available: http://enterprise-cn.netscout.com/enterprise-network/wireless-network/AirMagnet-WiFi-Analyzer
[25] J. Franklin, D. McCoy, and P. Tabriz, "Passive data link layer 802.11 wireless device driver fingerprinting," in *Proc. 15th USENIX Secur. Symp.*, 2006, pp. 167–178.
[26] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith, "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data," *SIAM J. Comput.*, vol. 38, no. 1, pp. 97–139, Jan. 2008.
[27] X. Li and M. Zhang, "Construction and security of fuzzy extractors," *J. Chongqing Univ. Technol. (Natural Sci.)*, vol. 24, no. 8, pp. 49–53, 2010.

AYONG YE (Member, IEEE) received the Ph.D. degree in computer system architecture from Xidian University, Xi'an. He is currently a Professor with the School of Mathematics and Information, Fujian Normal University. He is also the Vice Dean of the School of Network Security and Cryptology. He is also conducting information privacy and security research in the Fujian Provincial Key Laboratory of Network Security and Cryptology.

QING LI is currently pursuing the master's degree with Fujian Normal University, and conducting information privacy and security research in the Fujian Provincial Key Laboratory of Network Security and Cryptology.

QIANG ZHANG is currently pursuing the master's degree with Fujian Normal University, and conducting information privacy and security research in the Fujian Provincial Key Laboratory of Network Security and Cryptology.

BAORONG CHENG is currently pursuing the master's degree with Fujian Normal University, and conducting information privacy and security research in the Fujian Provincial Key Laboratory of Network Security and Cryptology.

• • •