ARSpy: Breaking Location-Based Multi-Player Augmented Reality Application for User Location Tracking

Jiacheng Shang, *Student Member, IEEE*, Si Chen, *Member, IEEE*, Jie Wu, *Fellow, IEEE*, and Shu Yin, *Member, IEEE*

Abstract—Augmented reality (AR) applications that overlay the perception of the real world with digitally generated information are on the cusp of commercial viability. AR has appeared in several commercial platforms like Microsoft HoloLens and smartphones. They extend the user experience beyond two dimensions and supplement the normal 3D world of a user. A typical location-based multi-player AR application works through a three-step process, wherein the system collects sensory data from the real world, identifies objects based on their context, and finally, renders information on top of senses of a user. However, because these AR applications frequently exchange data with users, they have exposed new individual and public safety issues. In this paper, we develop ARSpy, a user location tracking system solely based on network traffic information of the user, and we test it on location-based multi-player AR applications. We demonstrate the effectiveness and efficiency of the proposed scheme via real-world experiments on 12 volunteers and show that we could obtain the geolocation of any target with high accuracy. We also propose three mitigation methods to mitigate these side channel attacks. Our results reveal a potential security threat in current location-based multi-player AR applications and serve as a critical security reminder to a vast number of AR users.

Index Terms—Augmented reality, localization, attack

1 Introduction

Augmented reality (AR) applications connect the physical world and the cyber world by overlaying digitally generated information on the perception of the real world. Common AR applications use a *marker*, which is sufficient for AR projects where users can remain stationary, to trigger AR content. Location-based AR applications, in contrast, heavily rely on users' physical locations. Typically, they use GPS (BLE beacons for the indoor environment) and simultaneous localization and mapping (SLAM) techniques to determine the location of a user and to detect the orientation of a device. Utilizing location information to enhance an AR application helps to create a more immersive experience by relying on physical proximity to automatically trigger AR content. As the first significant success in location-based AR, okemon Go [3] of Niantic Lab, a smartphone game combing location-based real-time tracking and AR, attracted more than 45 million daily users within just a few days of its launch; it has been downloaded 800 million times since then. However, the potential of location-based AR lies far beyond smartphone games, and it is being applied more consequentially in both consumer and business-to-business settings. For example, Gatwick Airport has installed 2,000 indoor navigation beacons, which will enable AR path-finding at the airport [7]. Moreover, many third-party AR services such as Wikitude and Motive.io, provide a full-featured software development kit (SDK) that allows developers to build location-based AR applications without concern for technical details like motion tracking, proximity calculation, or scale estimation. In fact, with increasing shifts to hands-free devices, such as head-mounted displays or smart glasses, location-based AR is becoming a new information-delivery paradigm.

While the technology underlying AR applications is booming, little thought has been given to how these systems should protect the privacy of users. The AR devices continuously receive input from the environment through video, audio, and other sensors, and the continuous network connectivity will expose new security and privacy issues, especially in scenarios where AR users can also upload AR contents to the server (e.g., AR-based message board). Existing AR systems protect users' geolocations by encrypting the two-way transmission between users' devices and server using HyperText Transfer Protocol (HTTP) and HTTP Secure (HTTPS) protocols. Even if the attacker can capture the network packets in the middle of transmission, the geolocation of the user is regarded as safe if the attacker cannot decrypt the network packets. However, it is known that the attributes of encrypted traffic, often referred to as *side-channel information*, can leak some sensitive information about the communications. Such

- *Jiacheng Shang and Jie Wu are with the Department of Computer and Information Sciences, Temple University1805 N. Broad Street, Philadelphia, PA 19122. E-mail: {jiacheng.shang, jiewu}@temple.edu.*
- *Si Chen is with the Department of Computer Science, West Chester University of Pennsylvania25 University Ave, West Chester, PA 19383. E-mail: schen23@buffalo.edu.*
- *Shu Yin is with the School of Information Science and Technology, Shanghai Tech University, Shanghai 201210, China. E-mail: yinshu@shanghaitech.edu.cn.*

Manuscript received 10 Apr. 2020; revised 10 June 2020; accepted 26 June 2020. Date of publication 0 . 0000; date of current version 0 . 0000.
(Corresponding author: Jiacheng Shang.)
Digital Object Identifier no. 10.1109/TMC.2020.3007740

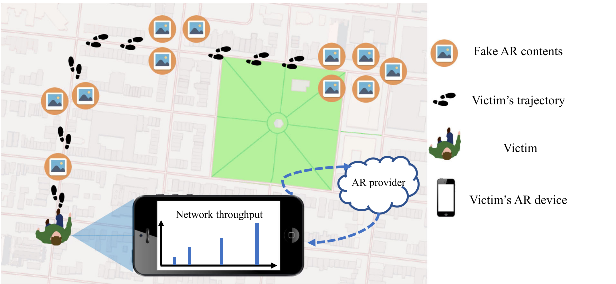


Fig. 1. An example show how the attacker infers the trajectory of the victim using network traffic.

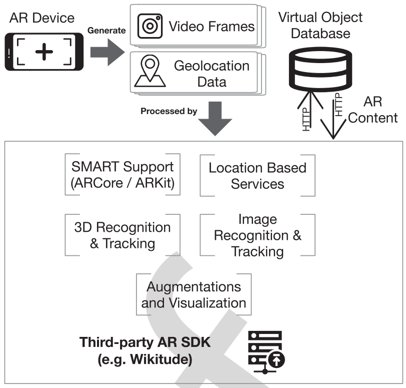


Fig. 2. A typical location-based AR application with third-party SDK.

side-channel information leaks have been studied by [44] (secure shell), [48] (voice-over-IP), and [10] (web application). Several existing studies conducted by various research groups have shown anonymity issues in encrypted web traffic. It has been shown that even when a user visits a web page through HTTPS channel, that page can still be identified due to the distinct size of a page and corresponding resource objects (e.g., images) [10]. Despite the importance of this side-channel threat in an encrypted channel, there is currently no study in the AR application domain for understanding its gravity and mitigation solutions.

In this work, we explore the security threat model of AR devices and demonstrate a new side-channel threat caused by location-based multi-player AR applications' unique combination of a high volume of real-time data, outsourced geolocation processing, and open privilege of uploading AR contents. We show that an adversary can covertly learn the location of an AR device and track the user in real-time by simply relying on monitoring the network throughput of the device. Different from getting GPS information from the device of the victim, the attacker can acquire network traffic information without using any location-related permissions, which means our attack methods are hard to be noticed by the victim in system permission level. Our attack model is proposed based on the following observations: 1) Location-based multi-player AR applications interact with a cloud database and cache AR contents when the victim is within a certain distance from them. 2) Many Location-based multi-player AR applications allow any user to upload or delete their AR contents to the database, such as WallaMe [5] and World Brush [6]. Therefore, as shown in Fig. 1, an attacker can also use the AR applications to upload fake AR contents of a specific size to the database in advance. Then, the attacker can observe a unique network traffic pattern on the AR device of the victim when the victim is close to that location. By properly determining the size and location of each AR content, the attacker can locate users and reconstruct the trajectory of the victim with high accuracy. Based on two observations, we propose a fake AR contents generation and deployment strategy. A network throughput processing method is also provided to extract the location information of the victim from the raw network throughput. Extensive experiments on our self-built AR application built on Android platform, simulation testbed, and a real location-based AR application show that our attack methods can reveal the location of the victim with high accuracy. Three mitigation solutions are also proposed to defend against this side-channel attack.

Our work makes the following contributions:

- We show that the network traffic information of an location-based AR applications can reveal potentially private location information.
- We propose strategies for generating and deploying fake AR contents in order to track the victim precisely. Also, we discuss the processing schemes of raw throughput data and the algorithm for reconstructing the trajectory of the victim.
- We implement our attack algorithms and build an automated user location tracking system. The real-world experiments on Android platform show that we could obtain the geolocation of any target with mean accuracy of at least 94.6 percent and perfectly reconstruct the trajectory of the victim with an accuracy of 77.5 percent in a small area. Moreover, our attack algorithms can infer at least top two locations with high accuracy of 86 percent based on a city-scale simulation.
- We discuss three potential mitigation methods to present this type of information leak in location-based AR applications and point out directions for a continuation of this work.

2 Preliminaries and Problem Formulation

2.1 Location-Based Multi-Player AR Overview

A typical location-based multi-player AR application runs on a mobile AR device. Users can utilize the equipped camera to record the surrounding real scene, combine the geolocation data from multiple sensors including GPS and gyroscope, and load the AR data information in real time. Then, they can make an integrated display of the acquired AR contents, such as texts, images, sounds, videos, and models. A typical location-based AR application structure is shown in Fig. 2. The sensor data (e.g., video and GPS information) is sent to the SDK-enabled logic layer of location-based AR applications. Location-based AR application processes the raw sensor data and requests corresponding AR contents from cloud dataset that is maintained by content providers. Then, the requested AR contents are download to location-based AR applications.

For location-based AR systems, the location-based AR contents are typically stored in a *cloud database* that is maintained by independent developers (e.g., *content provider*).

TABLE 1
Third-Party AR SDK Feature Comparison

	Geo API	GPS	Content API	Cloud API	Cost
ARCore	✓	✓	✓	✓(Cloud Anchors)	Free
ARkit2	✓	✓	✓	✓	Free
AR Studio	✓	✓	✓	✓	Free
ARcrowd	✓	✓	-	✓	Free + Commercial SDK option
ARmedia	✓	✓	-	-	Free + Commercial SDK option
ARPA	✓	-	-	-	Free + Commercial SDK option
Metaio SDK (now Apple inc)	✓	✓	✓	✓	Free + Commercial SDK option
DroidAR	✓	✓	OpenGL or jMonkey Engine	-	Free + Commercial SDK option
HoloBuilder	✓	✓	✓	✓	Free + Commercial SDK option
Kudan AR Engine	-	-	✓	-	Free + Commercial SDK option
Vuforia	-	✓	with Vuforia Cloud	✓	Free + Commercial SDK option
Wikitude	✓	✓	with Wikitude Studio and Cloud Recognition	✓	Free + Commercial SDK option
Motive.io	with Unity	✓	with Unity	-	Free +Commercial SDK option
EasyAR	-	-	-	-	Free

There are several reasons to move AR contents storage and geolocation processing to the cloud server. First, for business reasons, since the AR service mediates all AR content retrieval, the AR application developer can inject ads, charge content providers, and keep usage statistics easily. Second, to facilitate geolocation-based channel launching, recognition of trigger GPS location is done at the server, because this involves matching against proprietary databases using proprietary algorithms. Third, the geolocation contents are always considered as "hot" data, which keep changing all the time. The centralized location processing removes the need to replicate and update the geolocation content database on millions of devices, which is a computationally intensive task and would profoundly impact the actual performance of low-powered mobile devices.

Location-based AR applications are different from traditional location-based applications in terms of the content size. In general, the network traffic volume of location-based AR applications is much higher than most conventional location-based applications such as weather applications and navigation applications. Due to the large size of the AR contents, the AR applications only cache those AR contents that are within a certain distance from the AR user, which enables the attacker to estimate the location of the AR user by detecting a distinctive pattern in network traffic. Moreover, the network throughput of AR applications is much larger than that of traditional applications, and that is why 5G network is proposed to fulfill the network requirements of AR applications. In the AR scenarios, the large network throughput is much more normal than the traditional smartphone scenarios. This fact gives us a change to disguise our applications as an AR application that does not have location services, so that the network traffic introduced by fake AR contents cannot be easily noticed by the victim.

To support location-based multi-player AR experience, users can upload or delete their AR contents with real-world GPS coordinates to the cloud database and also download AR contents when they reach those real-world GPS coordinates. Moreover, location-based AR applications must continuously analyze the GPS location of the device in order to download AR contents at the GPS location and to anchor AR objects on the screen. Cognizant of the need to facilitate the development process, several AR *service providers* have supplied AR client software and SDK to the developers to help them build AR applications quickly, as listed in Table 1. We can see that most of them (except EasyAR) provide location-based (geo API or GPS) and cloud-based (content and cloud API) services to enable location-based multi-player AR experience. Moreover, most of them issue a free license, which means more developers will use these SDKs to build location-based AR applications. Therefore, without mitigation solutions in the SDK level, the location-based and cloud-based services can be used to infer the real-time location information.

2.2 Key Insight

Conceptually, a location-based AR application is quite similar to a traditional desktop application. They both work on input data from the user or the database, and their state-transitions are driven by their internal information flows (both data flows and control flows). The only fundamental difference between them is that an AR application's input points, program logic, and program states are split between the AR devices and the server, so a subset of its information flows must go through the network. We refer to them as data flows. Data flows are subject to eavesdropping on the wire and in the air, and thus often protected by HTTPS and Wi-Fi encryptions.

After the user submits the location to the server, the returned geolocation-based AR content is typically segmented at the application layer. In order to estimate the location of the victim based on the network traffic, the throughput patterns should always exist when a victim is walking along the path. Fig. 3 shows the downloading throughputs of every 10 seconds when the victim who uses

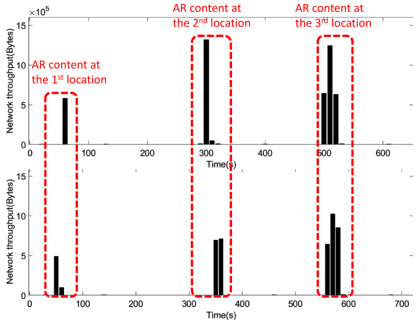


Fig. 3. Network throughputs of an AR application when a user walk along a path twice.

WallaMe walks along a path. On the path, 1, 2, and 4 AR contents (posts with pictures) are deployed at 3 different locations, respectively. Each burst represents a downloading job of AR contents when the victim reaches the location, and there is no significant network traffic between neighboring areas. We can see that the sizes of each burst and inter-burst intervals remain the same for the same AR content deployment at a different time. In fact, even if packets are encrypted using either the transport layer security (TLS) or secure sockets layer (SSL) protocol at the transport layer, their sizes and times of arrival are visible to the adversary. SSL/TLS is a separate protocol that inserts itself between the application protocol and the transport protocol (TCP) that enables applications to be only as secure as the underlying infrastructural components. This feature has been reported by many traffic analysis literature [35], [42]. Therefore, if the observable traffic feature is correlated with the segmentation in the application-layer, they can leak information about the content of the AR message.

The goal of attackers is to infer the geolocation information of the victim from the encrypted data traffic. In other words, an attack can be thought of as an ambiguity-set reduction process, where the ambiguity-set of a piece of data is the set containing all possible values of the data that are indistinguishable to the attacker. How effectively the attacker can reduce the size of the ambiguity-set quantifies the amount of information leaked out from the communications - if the ambiguity-set can be reduced to $1/R$ of its original size, we say that $log_2 R$ bits of entropy of the data are lost. Similar modeling of inference attack has also been discussed in prior research, e.g., elimination of impossible traces in [24].

2.3 Adversary Model

In this study, we consider a capability-restricted attacker that is aiming at revealing the location of users of a specific type of AR applications, called location-based multi-player AR. These AR applications allow users to publish or delete their own AR contents (e.g., images and messages) at any location. The capability of attackers is restricted in the following senses:

It only has the access right no more than that of a standard AR user (except that it can manipulate its geolocation). Manipulating geolocation is low-cost and easy to implement on many platforms. For example, Android allows users to manipulate location as long as developer options are activated. By manipulating its geolocation, the attacker can deploy AR contents using different accounts at any location without physically being there.

It can trick victims into installing its malicious applications that only require non-location-related permissions to monitor network throughput of the targeted AR application. In our adversary model, the attacker can only trick AR users into installing malicious applications that only require non-location-related permissions, which is a common assumption in side-channel attacks (e.g., the remote attack in [42]). There are two major ways to monitor the network throughput on current smartphone systems: 1) through internal system permission; 2) adding a virtual private network (VPN). On Android platform, the malicious application can get the network throughput of a specific application by using "read phone status and identity" permission that is widely required for many popular applications. Table 2 shows some popular applications that ask for "read phone status and identity" permission and their number of installation. We can observe that this permission is common, and it is hard for users to notice its potential risk of location leaking. Besides, the malicous application can also pretend as data usage monitoring applications that are popular on all platforms. For instance, My Data Manager [2] claims it is trusted by over 14.8 million uses worldwide on Apple Store. In general, these applications get network throughput by setting up a VPN. Any downloading data stream must pass the VPN before being received by an application. By using either of these two methods, the malicous application can get the real-time network throughput of the targeted AR application.

TABLE 2
Popular Mobile Applications That Ask for "Read Phone Status and Identity" Permission

Application	Number of installation
Tmall	1,000,000+
Youku	10,000,000+
Facebook	1,000,000,000+
Twitter	500,000,000+
Uber	100,000,000+

In summary, the location-based multi-player AR applications that may be used to track users' location must have the following features: 1) high volume of real-time data; 2) outsourced geolocation processing, and open privilege of uploading AR contents. Although such systems is only a small portion of all types AR applications, their users are enough to attract attackers to launch attacks.

3 Overview of the Attack

There are three parts to the location-based side channel attack: AR users (victim), AR cloud database, and malicious user (attacker). As shown in Fig. 4, a complete attack can be divided into five steps. 1). The attacker uploads several specially crafted geo-objects with a fake location to the cloud database. 2). The victim posts his/her current location to query the database. 3). The database returns several geo-objects back to the victim including the crafted objects. 4). The victim downloads these objects and creates a unique traffic pattern. 5). The attacker utilizes the malicious application to keep monitoring the traffic pattern of the victim and uses the reported pattern to reveal the location of the user.

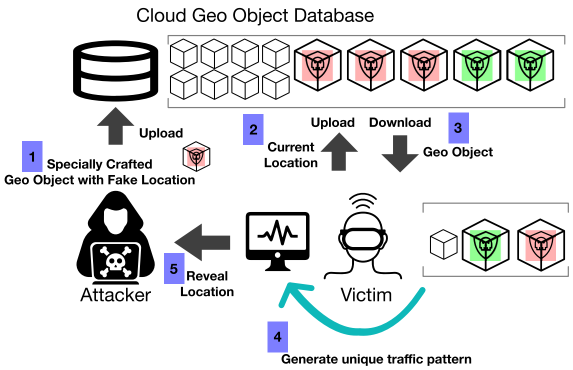
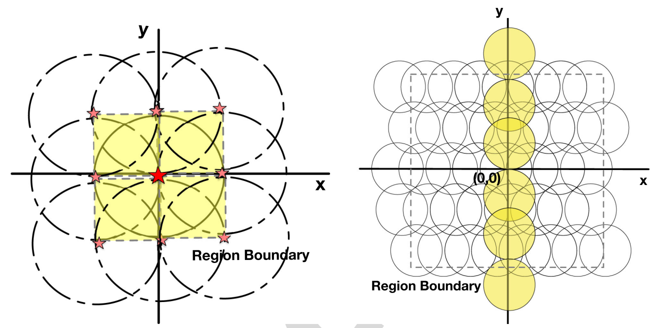


Fig. 4. Overview of the attack (in five steps).



(a) Illustration of space partition. (b) Strip-based space coverage strategy.

Fig. 5. Accuracy and coverage analysis for a 2D region.

3.1 AR Content Generation and Deployment

We first consider the simple attack scenario. In this scenario, the attacker already knows the small region where the victim is and wants to further infer the accurate location of the victim. We will discuss how to locate the victim in a large region in Section 3.2 for more general attack scenarios. To achieve this goal, we propose two AR content deployment strategies with different granularity and deployment cost.

AR Content Generation. There are two file formats that have been heavily adopted for displaying 3D models in AR: GL Transmission Format (glTF) and USDZ. Both are open-source format and can be generated from traditional 3D assets. The size of an AR content can be easily controlled by either adding a hidden surface or tweaking the image files (png format) that have been mapped to the 3D Model.

Coarse-Grained Location Detection. To locate the victim in a detected region, the basic idea is to cut the region into several non-overlapped areas. Each area is a circle whose center is the location of AR contents and radius is the searching range of the AR application. Moreover, the size of AR content in one area is distinct from that in any other areas. When a victim shows up in any area, corresponding AR contents will be downloaded to the device, and the attacker can infer the location of the victim based on the size of a downloading job in the network throughput. Although this strategy can locate the victim in an area with limited size of deployed AR contents, it has two key limitations. First, it cannot cover all locations in the small region since the searching area of each AR content is a circle. In some cases, victims in the region may not show up in the searching area of any AR content, so the coarse-grained location detection strategy fails to detect the location. Second, the localization granularity is relatively coarse. Without more information or deployment, we cannot infer more fine-grained location information of the victim within each non-overlapped searching area.

Fine-Grained Location Detection. To address the limitation of the course-grained location detection strategy, we also propose a fine-grained location detection strategy with more deployment cost to improve the localization granularity and coverage.

The location-based AR applications set a physical sensing range for each geo-content. For instance, in Pokemon Go, the AR content "Fort" is only reachable if the distance of the user is less than 38 meters. In order to further enhance the localization accuracy and thus break the limit, we utilize a space partition attack algorithm similar to [29]. The basic idea is to divide the target area into four non-overlapping regions and thus pinpoint the victim in the space to precisely one of the regions. Fig. 5a shows an example of the space partition. Assuming the covered area of an AR content is a box, given the maximum sensing range R, we can place fake AR contents at the origin (illustrated as red star in Fig. 5a) to cover a large area (highlighted in yellow). To improve the localization accuracy, the attacker can also place fake AR contents at four corners (illustrated as light red star) of the highlighted area. By doing this, the attacker can locate the victim in each smaller yellow box and further enhance the accuracy to $R/2$. We could repeat this partition for multiple rounds until the expected accuracy is achieved. The whole algorithm is summarized in Algorithm 1. For the simplicity of problem presentation, we consider the area where the victim is as the box rather than the circle.

Algorithm 1. Space Partition Algorithm for Fine-Grained Localization

In: Initial location I=(c_X, c_Y) and resolution δ
Out: Location set $\mathbf{P}$
1: Initial a queue $\mathbf{Q} \leftarrow (c_X, c_Y, \delta)$
2: **while** $\delta \geq$ *threshold* or $\mathbf{Q}$ is not empty **do**
3: $(c_X, c_Y, \delta) \leftarrow$ pop $\mathbf{Q}$
4: $\mathbf{P} \leftarrow (c_X \pm \delta, c_Y \pm \delta)$
5: $\mathbf{Q} \leftarrow (c_X \pm \delta/2, c_Y \pm \delta/2)$

We then study the case in which the small region is fully covered by the geo-AR content. We assume that each geo-AR content is capable of covering a fixed radius r around it. Therefore, we can model each geo-AR content as a disk with radius r. In order to cover the entire two-dimensional plan with these disks, the appropriate optimization metric should be the amount of geo-AR content used per unit area (e.g., density). We first introduce the strip-based deployment strategy (shown in the highlight part of Fig. 5b). The strip-based strategy places the geo-AR contents along a line such that the distance between the centers of any two adjacent circles is r. This strategy is good for tracking a user along a given path.

In order to tile the entire plane, we need to place the geo-AR content using the strip-based strategy repeatedly. Given a 2D plane, for every even index k, place a strip of geo-AR content oriented in parallel to the x-axis such that the point (0, $k(\frac{\sqrt{3}}{2}+1)r$) is the center of a geo-AR content constituting the strip. For every odd index k, place a strip of geo-AR content oriented parallel to the x-axis such that the point ($\frac{r}{2}$, $k(\frac{\sqrt{3}}{2}+1)r$) is the center of a geo-AR content in the strip. Next, we do a similar process along the y-axis. For every odd integer k, we place two geo-AR contents at (0, $k(\frac{\sqrt{3}}{2}+1)r \pm \frac{\sqrt{3}}{2}r$). The full geo-AR content displacement pattern is shown in Fig. 5b. It can be verified that our solution provides connected coverage to the entire two-dimensional region.

To support fine-grained localization with complete coverage, the key challenge is to propose a special AR content size sequence so that we can accurately locate the victim in any overlapped area. To address this issue, we design an AR content size generation algorithm based on *super increasing sequence*.

Let the sizes of crafted AR contents at different geolocations $\mathbf{W} = (w_1, w_2, \ldots, w_n)$ be a *super increasing sequence*. Then

$$w_k > w_{k-1} + \ldots + w_2 + w_1, \quad \text{for all } 2 \leq k \leq \text{ n}, \tag{1}$$

where each element w_i in set W is the size of AR contents deployed at a geolocation. Therefore, each AR content w_k has its unique size. Moreover, the combination of multiple AR contents is also unique. This property allows the attacker to place overlapped AR content, which greatly enhances the precision of our attack method. The size of each AR content w_i can be computed based on the Algorithm 2. Note that c is a constant value picked up by the attacker to avoid overflow. Once $\mathbf{W}$ is generated, the attacker can then execute an AR content generation function to generate a set of location-based AR contents based on the given size w_i. Note that this is an application-specific function, so the attacker may need to further alter the size (by adding or subtracting a constant value p) of each content or deployment multiple AR contents at a single location to achieve a successful attack based on the limitation of the AR application.

We can also notice that the fine-grained location detection is a special case of coarse-grained location detection. In coarse-grained location detection, each non-overlapped searching area must be a circle while each non-overlapped searching area can be in any shape. Although fine-grained location detection can achieve better granularity and coverage, it will also introduce more deployment cost since more non-overlapped searching areas are introduced. In real-world attack scenarios, the attacker can pick either strategy based on the trade-off between performance and cost.

Algorithm 2. AR Content Generator

In: Size n, Constant number c
Out: Set $\mathbf{W}$
1: **for** i `in range(1, n)` **do**
2: $\quad w_i \leftarrow sum(w_0, w_1, \ldots, w_{i-1}) + random(1, c)$

3.2 Recursive Region Detection

In real-world scenarios, it is usually hard for attackers to estimate the small region where the victim shows up. If we keep using proposed AR content deployment strategies for a large region, both of them will produce unlimited AR content size at some locations, which produces abnormal network traffic that the victim can easily notice and makes the attack unfeasible. In order to reduce the maximal size of the AR contents deployed at each location, we first narrow the search area by repeating partitioning a large area into four non-overlapped regions. For each partition, we deploy AR contents with the same size at all locations in each partitioned region, and the distance between neighboring AR contents is twice the length of the searching range of the AR application to avoid overlapped areas. To robustly distinguish four small regions based on the network throughput, four different sizes of AR contents for four different regions are generated based on Algorithm 2. Assuming the victim is moving, once a victim shows up in any region, our attack model can quickly identify the region of the victim. Then, our attack model deletes all AR contents and further repeats this process in the detected region until we can finally locate the victim within a much smaller region (e.g., a block) for further accurate localization and tracking. Fig. 6 shows an example of our hierarchical localization. The number of possible locations of the victim can be reduced to 4 after repeating the process twice.

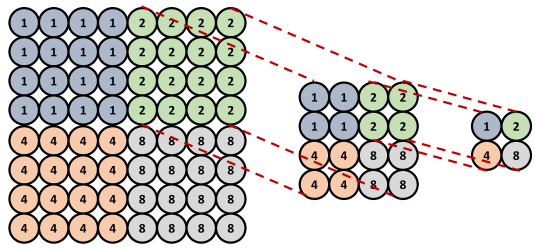


Fig. 6. Example of recursive region detection.

3.3 Network Throughput Processing

Noise Removal and Throughput Accumulation. The collected network traffic of AR applications contains noise. On the one hand, the noise comes from various link conditions or other data exchange except downloading AR contents between the AR application and the server. On the other hand, based on our experiments, the network throughput not only counts the bytes of the content in packets but also counts the bytes in packet header or other information within the packets. So, the raw network traffic data cannot be directly used to parse the real-time locations. In order to accurately track the location of the victim using network traffic, we first eliminate small traffic that cannot be caused due to downloading AR contents from the server based on a threshold τ. Moreover, we need to accurately estimate the network throughput downloaded at each location in order to infer the location of the victim based on the special throughput pattern. Since the AR contents are not downloaded immediately, we need to accumulate the network throughput within a moving time window of length T in order to accurately estimate the size of AR contents deployed at each location. The length of the moving time window is set as

$$T = \left\lfloor \frac{max(\mathbf{W})}{\lambda} * Fs \right\rfloor + 1, \tag{2}$$

where $max(\mathbf{W})$ is the maximal size of AR contents, λ is the average downloading speed of the AR application measured

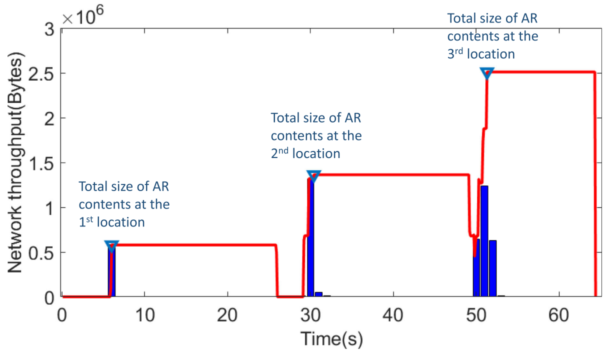


Fig. 7. Noise removal and throughput accumulation.

based on the history, and Fs is the sampling rate of network traffic monitoring. As we observe in Section 2.2, there is no significant network traffic between two neighboring locations, which means that the size of AR contents at each location is a local maxima in accumulated throughput sequence. Based on this observation, we find the size of AR contents by finding the local maxima in the accumulated throughput sequence. Each local maxima represents a single location where the AR contents are deployed or a location that is within the searching range of AR contents deployed at multiple locations. Fig. 7 shows an example of our network traffic processing. The raw network throughput is collected from WallaMe when the victim passes three locations where 1, 2, and 3 pictures are deployed respectively. After throughput accumulation, we can observe three local maximas (blue markers) in accumulated throughput that correspond to three downloading jobs in raw network throughput.

Algorithm 3. Localization Algorithm

In: A local maxima in accumulated throughput sequence $\mathbf{S}$, generated AR content size set $\mathbf{W}$
Out: Inferred location $\mathbf{X}$

```
X = ∅
n ← sizeOf(W)
for i in range(n, 1) do
  if S > w_i then
    X = X ∪ x_i
    S ← S − w_i
  else
    x_i ← 0
```

Localization.

The localization algorithm works as follows: given a local maxima $\mathbf{S}$ in accumulated throughput sequence and generated AR content size set $\mathbf{W}$, we aim to infer the location $\mathbf{X} = (x_1, x_2, \ldots, x_m)$ of the victim. $\mathbf{X} = (x_1, x_2, \ldots, x_m)$ means the location that is within the searching range of AR contents deployed at m different locations, and m is a positive integer. For the coarse-grained strategy, $m = 1$. For the fine-grained strategy, since the integers in set $\mathbf{W}$ form a *super increasing sequence*, we can prove that the inferred area $\mathbf{X} = (x_1, x_2, \ldots, x_m)$ is unique if $\mathbf{X}$ exists. The location $\mathbf{X}$ can be computed by the following localization algorithm (Algorithm 3).

After obtaining the location of each local maxima, we further calibrate the localization results. If the victim is detected in the overlapped area of a set of locations, we will double check the physical distance among locations in the set. If the searching ranges of locations in the set do not have a common overlapped area, we argue that the received throughput is noisy and the victim is at the feasible overlapped area of a subset of locations whose total size is maximal. The trajectory of the victim is recovered once all of its locations on the path are obtained. However, due to inaccurate GPS data, a victim may receive the data that is deployed by the attacker more than one time. To remove redundant information, for a sequence of continuous and identical location estimations, we only reserve one of them.

4 Experimental Setup

4.1 Implementation

We built a real testbed in order to effectively evaluate the attack methods we propose. Our testbed included four parts: an Android application for monitoring network traffic, an Android application for imitating the behaviors of current AR applications, a customized location provider for location spoofing, and a back-end server that receives the requests from all AR clients and returns corresponding data. Simple graphical user interfaces (GUI) are designed to help subjects to collect data. We illustrate the system design and implementation in detail in the following paragraphs.

Network Traffic Monitoring. The core part of our system is accurately monitoring the network traffic of a specific application. To achieve this goal, we studied the feasibility of monitoring network traffic on Android platform. NetworkStatsManager can provide access to network usage history and statistics of other applications, which enables an attacker to implement a listener in another application on a device of the victim. Although NetworkStatsManager needs "read phone status and identity" permission that is a protected permission, a lot of popular Android applications ask for this permission, as shown in Table 2. This fact enables the attacker to hide this listener in a popular application without being noticed by the victim. To get the real-time network, we created a background service that can log the total network usage every second. The throughput of each second was acquired by calculating the difference between neighboring entries in the log file.

Location Spoofing. Location spoofing is used to generate mock locations, so that the attack can deploy fake AR contents at any location without physically being there. Moreover, other AR users can also deploy AR contents, which may change the pattern of network traffic on the device of the victim and break our attack model. To address this problem, the attacker needs to know the size of AR contents deployed by AR users at those locations the victim may appear, which can also be solved by leveraging location spoofing. Before deploying fake AR contents, the attacker first sends fake geographical locations where he/she wants to deploy fake AR contents to the server. The attacker monitors the network traffic that reflects the size of AR contents deployed by normal AR users. Based on the size of existing AR contents, the attacker rearranges the size of fake AR contents deployed at each location. Most of the Android applications acquire location via a location provider (e.g., "network" or "GPS") from the location system service. However, it is possible to add customized location providers under certain circumstances such as debugging. The attacker needs to enable "Allow mock location" option in the developer options of their Android device before getting access to the mock location API. This API asks for five variables: *latitude, longitude, altitude, speed* and *accuracy*. Typically, the location-based AR applications only utilize the *latitude* and *longitude* values to determine current location of the victim.

Location-Based AR Application. We first studied several state-of-the-art AR applications and SDKs (e.g., Google AR and Wikitude) and found that these AR applications and SDKs send local information (e.g., locations and images) to the server using a simple HTTP(S) GET requests. After getting the requests from the client, the AR server serializes returned information into a structured data (e.g., JavaScript Object Notation, Extensible Markup Language, and Protocol Buffers) using HTTP protocol and returns it to the AR application. Extended studies show that all existing AR applications and AR SDKs are based on the same mechanism. Therefore, our AR application is equivalent to most existing applications or future applications developed using current SDKs in terms of data transmission and communication.

Therefore, we built a location-based Android application to imitate the behavior of current AR applications. The application keeps collecting GPS information, sends it to our back-end server using GET request, and receives corresponding data from the server. We further tested it and ensured that our AR application has the same behavior of network traffic and mechanism for data transmission. To ensure the GPS locations sent to the server are accurate, we only sent a GET request to the server when the accuracy of measured GPS data was within 8 meters. Although our self-built AR application had most features of real AR applications, we could not perfectly simulate and reproduce all behaviors of real AR applications. To further show that our attack model is feasible to be launched on real AR applications, we also evaluated our attack model on WallaMe.

Back-End Server. We implemented our server on a public IP address based on HTTP(S) protocol. The back-end server receives requests from all mobile clients, analyzes their geographical location, and returns corresponding AR contents. For each request, we first compared its GPS location to those of all deployed AR contents. If the user appeared around one or more AR contents, we would generate a temporal file with the corresponding size in milliseconds and return it in the response.

4.2 Data Collection

To evaluate the performance of our attack model, we conducted various experiments on our testbed on a campus, as shown in Fig. 8. On the path, we uniformly picked 8 locations on the map. The distance between neighboring locations was about 60 meters. The searching range of each location was set to different values to evaluate the performance of our attack strategies on detecting single location and detecting the overlapped area. For coarse-grained location detection, the searching radius is about 20 meters, and the size of AR contents at the first location was set to 1 KB and was increased by 1 KB for each of the following locations. For fine-grained location detection, the searching radius is about 45 meters, and we deployed AR contents whose total sizes follow the rule of *super increasing sequence* at each location. The minimal size of deployed AR contents was also 1 KB. We control the size of AR contents at each location by 2 ways: 1). Deploying more AR contents with equal size. 2). Changing the size of a single AR content by adding more information (e.g., pictures with specific sizes) to the content. 12 healthy volunteers with their ages ranging from 21 to 26 were involved in the study. Among 12 volunteers, we asked 8 of them to use our self-built AR application and the other 4 of them to use WallaMe for testing. We collected 10 trials from each volunteer, and each trial lasted for about 10 minutes. During each trial, the volunteer was required to pick a path and pass different locations while opening two applications (the AR application and the network monitoring application) and connecting their devices to their personal hotspot. The server returned corresponding AR contents based on the real-time location of the user without introducing extra traffic. Besides recording the network traffic of the AR application during each trial, we also logged the received GPS coordinates as the ground truth.

Fig. 8. The AR contents deployment.

4.3 Evaluation Metric

The location reported by our system is not the real geolocation with 2-dimension coordinates but a non-overlapped area. The size of the reported area is determined how the attacker perform coarse-grained and fine-grained location detection. Therefore, instead of using the distance as a metric, we evaluate the system performance based on how accurately our system can correctly locate the victim in a area. Here a correct detection means that our attack system detect that the user is in an area when the user is exactly in that area. The location detection accuracy $Accu$ is defined as

$$Accu = L_{correct}/L_{all}, \tag{3}$$

where $L_{correct}$ is number of correctly detected location (area) and L_{all} is the number of all locations (areas) that the victim has passed.

5 Experimental Results

5.1 Performance of Location Detection

Single Location (Non-Overlapped Area) Detection. Since the performances of both the recursive region detection and localization strategies are based on how accurately we can detect the victim at a location, we first evaluated the

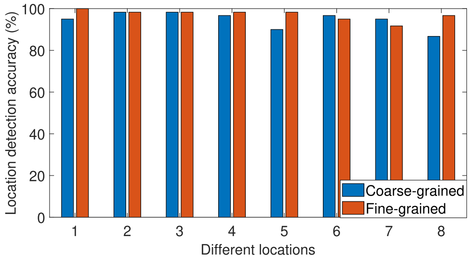


Fig. 9. Performance of a single location detection.

performance of our attack model on the single location detection using our self-built AR application. We removed the noise in the raw data and processed it with our location detection algorithm. Then, we compared the detection results with the ground truth. The location detection accuracy is defined as the number of correct detections divided by the total number of detections.

Fig. 9 shows the single locations detection accuracies for all locations by using either the coarse-grained location detection strategy or the fine-grained location detection strategy. Evaluation results show that our coarse-grained location detection strategy can locate the victim in non-overlapped areas with a mean accuracy of about 94.6 percent. Since the size differences of AR contents deployed at different locations are much greater in fine-grained strategy by using *super increasing sequence*, it can provide a better average accuracy of about 97.1 percent. Moreover, we notice that the location detection accuracies are slightly lower for those locations where more AR contents were deployed. The reason is that the downloading process of large files is easier to be influenced by unstable network, which breaks the special patterns in network throughput.

Overlapped Area Detection. In our fine-grained location detection strategy, the victim may also appear in the searching areas of AR contents deployed at multiple locations. Instead of assigning the victim to one location, we would like to locate the victim in the overlapped area accurately. To evaluate how accurately our system can detect the overlapped area, we evaluated how accurately the victim can be located in the overlapped area of two searching areas. Since the sizes of AR contents deployed by the attacker are unique at different locations, one location should have more AR contents deployed than the other one. We repeated the experiment for 100 times, and Table 3 shows the detection accuracy for the location with more AR contents and the location with less AR contents. We can see that our location detection model can accurately detect the location with more AR content, and the percentage of the wrong prediction for the location with less AR contents is no more than 1 percent. These results show that it is feasible to locate the victim even if he/she is in an overlapped area.

Granularity of Location Detection. There is a trade-off in how densely the attacker should deployed the AR contents in a small region. If we deploy AR contents at many different locations, we can estimate locations of the victim with a better granularity, but the location detection accuracy may not be good due to inaccurate GPS coordinates. Also, the network usage is also higher, which makes our attack easy to be noticed by the victim. In fact, the fewer locations where we deploy AR contents, the better the detection accuracy is expected to be, but more details of the trajectory of the victim are lost. In order to study how densely the AR contents can be deployed with a good detection accuracy in our attack model, we adjusted the distance between neighboring locations and studied its influences on location detection using our self-build testbed. In this experiment, the searching range of AR contents at each location was about 20 meters. We asked a volunteer to walk along the same path for 10 times. Along the path, we deployed AR contents at as many locations as possible with the distances between neighboring locations were about 70 meters, 27 meters, and 13 meters, and the results are shown in Table 4. When the distance is larger than 27 meters, ARSpy can achieve an excellent location detection accuracy of at least 98 percent since at most 14 percent of the searching area is overlapped with those of neighboring locations. The accuracy drops to 60 percent when the distance is about 13 meters. Considering the deviation of GPS measurements is from 3 meters to 8 meters in outdoor environment, the noisy GPS data cannot reflect the real-time location of the victim relative to each location where AR contents were deployed. If GPS data is inaccurate, the server would not consider the victim is at that location. Therefore, the device of the victim would not download the AR contents deployed by the attacker, and the attacker cannot track the victim based on the network throughput.

TABLE 3
Location Detection Accuracy for the Overlapped Area

Location with	More AR content	Less AR contents
Accuracy	100%	99%

TABLE 4
Location-Detection Accuracies With Different Distances Between Neighboring Locations

Distance (meter)	70	27	13
Accuracy	100%	98%	60%

5.2 Performance of Trajectory Construction

It may also be beneficial for the attacker to know the actual route through which the victim traverses on his way to the destination. For this purpose, we also calculate for each constructed trajectory the Levenshtein distance [28] between it and the actual trajectory. The Levenshtein distance is a standard metric for measuring the difference between two sequences. It equals the minimum number of updates required to change one sequence to the next. The distance is normalized by the length of the longer trajectory of the two. This allows us to measure the accuracy of the algorithm for estimating the full trajectory the user traversed. For each estimation, we also note whether it is an exact fit with the actual route (i.e., zero distance). The percentage of successful localization of destination, average Levenshtein distance, and percentage of exact full route fits are calculated for each type of estimated route. To benchmark the results, we note in each table the performance of a random guess algorithm which outputs merely a random but feasible route.

TABLE 5
Performance of Trajectory Construction

	Destination	Avg Levenshtein distance	Exact fit
ARSpy	96.72%	0.0345	77.5%
RG	13.75%	0.7665	0

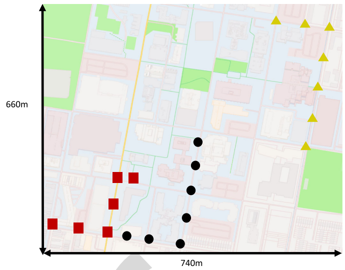
We evaluate the performance of the trajectory construction using the dataset for single location detection, and Table 5 illustrates the trajectory construction performance of ARSpy and the random guess (RG)-based attack. Compared with the random guess-based attack model, ARSpy can achieve much better trajectory construction performance. Moreover, ARSpy can accurately predict the destination of the victim with a high accuracy of 96.72 percent. Although the percentage of exact full route fits is 77.5 percent, we can note that the average normalized Levenshtein distance is only 0.0345, which means only one or two locations are wrongly detected for a path with eight locations even if the constructed trajectory does not fit the real trajectory. The high percentage of successful localization of destination and the low percentage of exact full route fits show that ARSpy can accurately track the victim for a more prolonged path.

5.3 Performance on A Real AR Application

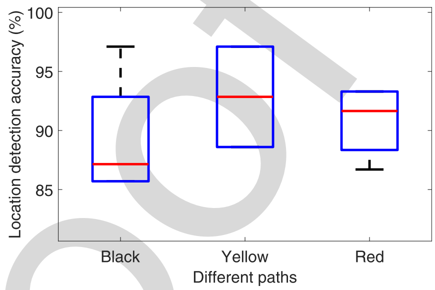
Experimental results show that our attack methods can achieve high performance on our self-built AR application. To show the feasibility of our attack methods on a large range of real AR applications, we evaluated the performance on WallaMe. WallaMe is a free AR application that allows users to take a picture of a surface around them and add information (e.g., words, stickers, and photos) on them. Once the picture is posted, it will be geolocalized and visible by everyone passing by. Also, the user who uploads the picture can make the pictures private, which means that the picture is visible only to specific groups of people. In this experiment, we evaluated how accurately three locations can be detected by our attack methods. Specifically, we deployed 1, 2, and 4 pictures at three locations, and the sizes of those pictures are nearly equal. Since the searching radius of WallaMe is about 1 block, the distance between each possible next location and the current location is about 180 meters to generate overlapped areas between two neighboring locations. We ask four volunteers to pass these three locations for 20 times. Based on the accuracies of GPS measurements, each volunteer was regarded at a single location or the overlapped area of two locations by WallaMe. Experimental results show that our system can successfully detect the single location or the overlapped area with an average accuracy of 90 percent, which implies that our attack model is also feasible to be launched on existing AR applications.

5.4 Influence of Different Paths

To show the generalization of our attack model, we further evaluated the location detection accuracy for three other paths. As shown in Fig. 10a, a volunteer was asked to walk through each region along the colored path. The three paths were carefully selected to cover different types of outdoor scenarios. For example, the regions of the black path and the yellow path have high buildings of at least eight floors. We used these two regions to evaluate the attack performance with inaccurate GPS measurement due to high buildings. The region of the red path has low building of at most four floors. Across all experiments in this subsection, we used our coarse-grained location detection strategy to deploy AR contents at different locations. The distance between neighboring locations is about 50 meters, and the searching radius of each location is set as 20 meters. The experimental results are shown in Fig. 10b. We can see that our system can provide location detection accuracy for at least 85.7 percent even if the GPS measurements are influenced by the high buildings.



(a) Deployment of AR contents at three paths.



(b) Average location detection accuracy.

Fig. 10. System performance for three paths.

5.5 Performance on Different Devices

In our attack model, we assume that the attacker does not know what smartphone the victim uses. Therefore, we further conducted experiments to evaluate the effectiveness of our attack model on different devices using the deployment configuration in Fig. 8 using the coarse-grained strategy. We chose Google Nexus 5 and Nexus 6 as devices in this experiment. During the experiment, we asked a volunteer to hold two devices while walking along the path for ten times. Fig. 11 shows the location detection accuracy on two devices. We found that Nexus 6 has a better performance than Nexus 5. The reason is the sampling rate of GPS data. In most cases, both of the two devices can receive a GPS coordinate every 1 second. However, Nexus 5 needs to wait for more than 1 second to get the next GPS coordinate at a probability of 1.39 percent in our experiments, while Nexus 6 only has this issue at a probability of 0.93 percent. Moreover, the maximal delay of receiving the next GPS coordinate on Nexus 5 was 22 seconds, while that on Nexus 6 was only 12 seconds. Considering the walking speed is about $1.4\ \mathrm{m/s}$, a victim using Nexus 5 is more likely to miss a

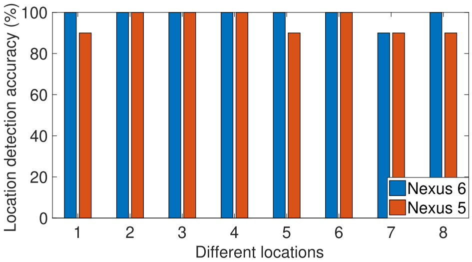


Fig. 11. Location-detection accuracies on two devices.

location due to the long delay. Even if the device of the victim may lose some GPS coordinates, our attack model can achieve high location detection accuracy on both of the two devices, and at most one location was missed in each trial. The high location detection accuracy indicates that our attack model is feasible on different devices, which means that the attacker can deploy this attack on any victim who is using AR applications.

5.6 Performance on Large-Scale Long-Term Tracking Simulation

According to [50], the top N locations inferred from human mobility data can be used to reveal the identity of a user. For instance, top two locations may link to home and work locations of the user, top three locations may correspond to home, work and shopping locations. [15] shows that the human mobility traces are highly unique and more than 95 percent of the individuals could be uniquely identified based on the top four locations. In this subsection, we discuss the performance on long-term tracking of the top N locations inferred by the network throughput data of the user. The simulation is based on the GeoLife Dataset [51], which contains GPS trajectories of 182 users in a period of over three years. We replayed the GPS data (as ground truth) to simulate user moving trajectory in a location-based AR application that we created and then used ARSpy system to launch attack and infer locations of the user. Our simulation assumes that the attacker has some pre-knowledge of the target, and knows the city that he/she lives in. We set the detection range of each AR content equal to 1.2 km. Shown in Fig. 12, our AR attack method is able to deduce at least top four locations for more than 50 percent the user data and achieves 86 percent detection rate for the top two (and above) locations. This means that the attacker can infer these users' home and workplace solely based on the network throughput data.

Fig. 13 illustrates the relationship between the number of deployed AR contents and the margin of location error. In this experiment, we calculate the distance between the location reported by our AR attack method and the ground

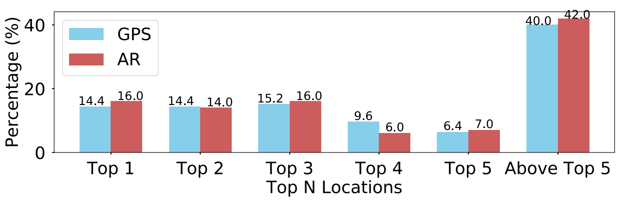


Fig. 12. Percentage of top N locations.

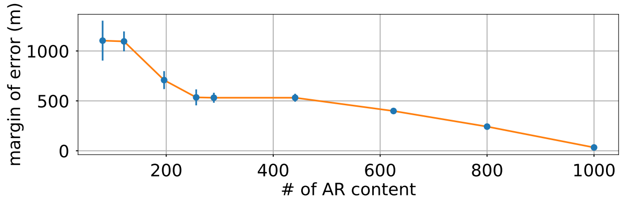


Fig. 13. Margin of error verses number of AR contents.

truth. According to the results, to track an individual user in a city, the attacker needs to deploy at least 1,000 fake AR contents to the server to bring down the error to around 30 meters. However, the actual number depends on the detection range of the AR contents and the size of the tracking area. The results show that our proposed algorithms can be used for long term tracking and is able to correctly infer top N locations of the user with high accuracy, which means that the attacker can track a target even if the server puts a restriction on the AR content update rate.

5.7 Influence of Traffic Noise

The logged throughputs always contain noise. The noise can be from other applications on the same device. For example, the downloading jobs of other applications will cause network congestion, which may change the traffic pattern of the AR application and make it difficult for the attacker to recognize deployed AR contents. On the other hand, the noise can also be from other downloading jobs generated within the same AR application. For example, the AR application needs to synchronize with the server and download relative contents. This kind of traffic can be wrongly recognized as AR content deployed by the attacker, which leads to the attacker being unable to construct the trajectory of the victim.

To evaluate the influence of downloading jobs of other applications, we let a volunteer walk along the path in Fig. 8 for 10 times while using our self-built AR application. At the same time, the volunteer downloads a large file via Google Play on the same device. Experiment results show that all locations can be detected, which implies that the downloading jobs of other applications will not destroy the traffic pattern of AR application and thus do not influence the location detection performance of our attack model.

In order to evaluate the location detection performance under the influence of extra traffic generated by the AR application, we let the server send extra data to our AR application based on the network throughput distribution of Ingress and Pokemon. In this experiment, the threshold τ is set to the minimal size of fake AR contents in order to remove the influence of extra network traffic. Since *super increasing sequence* determines the size of AR content deployed at each location, the smallest size of AR contents at all locations should be as small as possible. We set the smallest size to different values in order to evaluate what is the smallest size of AR contents required to ensure good location detection accuracy. Experimental results show that the location detection accuracy rises with the increase in size of fake AR contents. When the size of the smallest fake AR content is 20 KB, ARSpy can provide location detection accuracy of at least 92 percent, which proves that our attack

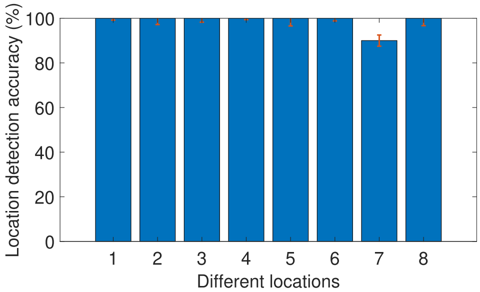


Fig. 14. Location detection accuracies in jogging scenario.

model is feasible even if the AR application frequently exchanges extra data with the server.

5.8 Influence of Different Speeds

In Sections 5.1 and 5.2, we only evaluate the performance of our attack model in the scenario where the victim is walking. In this subsection, we evaluate the location detection performance of our attack model when the victim is jogging at a speed of about $2.5\ \mathrm{m/s}$. Here we do not consider the scenario where the victim is running at high speed since more network bandwidth and computation resources are required in this scenario, and few AR applications are designed for running. In this experiment, we ask a subject to run two applications we build while jogging along the path on the campus. Experiment results are shown in Fig. 14. It is clear that our attack model can achieve almost 100 percent location detection accuracy for all 8 locations. By comparing the ground truth in both walking and jogging scenarios, we find that the trajectory in jogging scenario can be approximately produced by decimating the trajectory in a walking scenario by 2. As long as the searching radius (12 meters in our system) can tolerate the displacement of the victim between two GPS measurements (about 5 meters while jogging), our scheme can still track the victim even if the victim is moving at high speed.

5.9 Battery Consumption

Besides accuracy and robustness, battery consumption is another important issue we need to consider when performing an attack. If an attack model requires a significant portion of available CPU time, the significant battery drain can be quickly noticed by the victim. Current security solutions can detect a variety of attacks by sensing abnormal battery behavior and energy patterns [18]. In our attack model, the network traffic monitoring is the key way to perform the attack and may cause battery drain. In order to evaluate the battery consumption of our network traffic monitoring, we used Batterystats and Battery Historian [1] to collect battery data. Battery Historian converted the report from Batterystats into an HTML visualization in the browser and provided the battery data in a process level. During the experiment, we ran the application for about 75 minutes while all the other applications on the target smartphone remained closed and the screen kept on. Experimental results show that our network traffic monitoring application consumed about 0.03 percent of the total energy, while the battery consumption of GOOGLE_SERVICE was 0.05 percent. The results show that our attack model only introduces insignificant battery drain that cannot be detected by the victim and the battery behavior-based security solutions.

6 Mitigations

The cause of privacy leaks in location-based AR application is that, for the same path, the network throughput changes over the time is in a unique and identifiable way. Segmenting the returned data may reduce the granularity of the leak but does not prevent the attacker from revealing the location. A straightforward solution would be to store all AR contents locally (like Pokemon Go) to totally eliminate the potential information leak since the AR applications do not download contents from the server in real time. However, this solution is not suitable for large-scale AR systems which contain tons of ever-changing 3D AR models. Another solution would be padding each packet to achieve constant-size encoding to eliminate the leak, at the risk of a very inefficient encoding scheme since it would require transmitting more redundant traffic than the actual content size. Similarly, implementing a tight rate control mechanism would result in an inefficient transmission protocol.

In order to limit the capability of the attacker, we propose 3 possible mitigation methods. First, SDK providers or developers can deploy and maintain an active cache with variable size to store the AR contents on the client side. The AR contents can be not only downloaded to the cache when the AR user reaches the location but also prefetched from the server based on the location of the victim and movement pattern. Once the AR contents are prefetched to the cache, they can be enabled to be displayed by sending a control signal instead of completely re-downloading it. Thus, the attacker has to know the detailed implementation of such variable cache and predict the movement of the victim in the same way as the server. Otherwise, the network throughput pattern can be destroyed, and the attacker cannot reconstruct the trajectory of the victim.

Second, the developers of AR application can put more limitations on AR users. For example, any AR user cannot deploy too many AR contents at a single location. Meanwhile, the size of each AR contents should not be too large. By doing this, the capability of the attacker is greatly limited since the property of *super increasing sequence* is hard to be satisfied. For example, the general deployment strategy cannot work since the maximal number of AR contents is limited.

Another method is to further limit the permission of network traffic monitoring on the devices of the victim, which means third-party applications cannot get the network traffic information. Existing mobile operating systems have noticed the potential threat of exposure of network traffic information. For example, Android protects network traffic information using "read phone status and identity" permission, but the user can still be deceived to install malicious applications since many popular applications also ask for this permission. Similarly, network content filters are not permitted for regular applications in Apple store, but the attacker can disguise the malicious application as a normal application (e.g., Sift [4]) and deceive users to install the malicious application on the device. To address this issue, the network traffic data should be visible only to the operating systems, and the users should be alerted if the network traffic information is being monitored by any service.

7 Discussion

Scaling Our Approaches to Various AR Applications. Different AR applications may use different compressing algorithms, which results in different traffic patterns for the same AR content and influences the geolocation estimation. However, as long as the developers do not change the way to store and deliver the AR contents, the attacker can still know the relationship between the size of fake AR content and the traffic pattern after collecting enough data of a new AR application.

Influence of Other AR Contents From Other Users. In real scenarios, there may also be AR contents from other users around some geolocations, which alters the traffic pattern on the device of the victim. The attacker can address this issue by monitoring the size of all AR contents at each geolocation periodically. If some AR contents are already deployed by other users at a particular location, the attacker can change the size of fake AR contents accordingly so that the total size of fake and normal AR contents at each geolocation meets the requirement of either coarse-grained or fine-grained localization. The attacker can dynamiclly change the cycle time based on the size of interested region and cost. Even if other users frequently change the network traffic profile at a locations, the attackers can give up the current attack and restart attacking the victim when the victim reaches a better area. Although the attackers can lose much location information of the victim, but these limited information can still be aggregated with other data to infer more locations of the victim that are not detected by our system. For instance, [15] shows that the human mobility traces are highly unique and more than 95 percent of the individuals could be uniquely identified based on the top four locations. Therefore, the other locations of the victim in a day can be easily inferred by combining our detection results with other anonymous location dataset.

Limitations and Future Work. Our system involved a limited number of participants, and all users are university students. To better understand the performance of our system, more participants with more diverse backgrounds must be engaged. Also, the experiments were all conducted within 6 months. Considering that human behaviors and habits (e.g., spe-ed of walking) may change, a long-term evaluation should be conducted. Besides, we only used WallaMe as an example to show the effectiveness of our system on real AR applications. In the future, we plan to evaluate our system for more AR applications and study how behaviors of different AR applications influence the performance of our attack model.

In our testbed, considering most location-based AR applications are designed for outdoor scenarios, we only tested our system in outdoor environments. In future work, we plan to implement our system for indoor AR applications that has indoor localization and computer vision techniques. Moreover, we will study using machine-learning techniques to improve the accuracy and robustness of location detection for real AR applications.

8 Related Work

Mobile Augmented Reality. The basic idea of augmented reality was proposed in the 1960s [9], [45]. Since the 1990s, researchers have become increasingly interested in this area, and many AR devices and frameworks have been proposed to overcome challenges to tracking and registration in the hopes of properly aligning virtual and real objects, user interfaces and human factors, and auxiliary sensing devices. The increasing capabilities of mobile devices, affordable high-speed Internet access, and breakthroughs in computer vision and cloud computing have only recently made AR a reality. Many mobile augmented reality (MAR) applications have been designed and implemented towards the following demands: 1). Tourism and navigation [12], [19], [20], [21]; 2). Advertisement [23], [34]; and 3). Entertainment [38]. In [20], [21], researchers propose a MAR prototype for campus exploration. The application can display information about surroundings while users are walking.

Augmented Reality Security. Lately, several researchers have focused on the security, privacy and safety concerns associated with AR system [14], [25], [43]. However, most of the existing publications are focused either on *input privacy* [22], [37], [41] or *output safety* [25], [26], [27]. Only a few publications [11], [40], [46] are addressing *output privacy* of AR system. Different from existing works, we point out a novel side channel that allows attacker to track an AR user even if the network traffic is encrypted.

Fingerprinting and Traffic Analysis. There is a large body of research on the side-channel attack on encrypted network traffic for traditional website [8], [36], [42]. In [36], the authors evaluate a state-of-the-art method for detecting a website and conclude that webpage detection is infeasible. X. Cai *et al.* [8] proposed an attack method that can guess which of 100 web pages a victim was visiting with an accuracy of at least 50 percent. A more recent work [42] shows that it is possible to identify encrypted video streams in high precision. Besides website information, traffic analysis can also be used to infer application-specific sensitive information, such as health conditions [33], or other contextual information [13]. A recent work [32] is proposed to detect AR users' locations by monitoring the network throughput. However, their solution only considers an small area (three locations) and involves much training cost. Prior works also cover mitigations [31], [49] and counter-mitigations [17].

Location Leakage Through Sensory Data. In the past a few years, researchers did a lot of works on inferring locations using various types of sensory data and side channel information [16], [30], [39], [47]. For example, Liang *et al.* proposed a system to infer the locations of a user using motion sensors [30]. However, their system requires pre-collecting enough training data from the same user for the same path. Therefore, their system can fail to work as long as the user change the movement behavior. Besides using sensory data from a single source, researchers also seek to predict the next location of a user using multiple sensors and context information. For instance, Do *et al.* try to predict the next location of the user using current context consisting of current location, time, application usage, and etc. [16]. However, such a model can only work when the behaviors of the user is relatively stable. To reduce the impact of dynamic behaviors of a user, Tiwari *et al.* design an attack model that can infer location-related information of a user using the network traffic when the user is using Google Map [47]. However, they can only provide good performance on path detection over the time while fail to detect the real-time location of a user. Compared with existing work, our system does not need to collect any training data from the target user. In addition, our attack model does not rely on any consumption on the behaviors of the victim. Moreover, compared with existing works that also leverage network traffic, our system is specifinally designed for AR applications and can achieve better system performance on single location detection in real time.

9 Conclusion

The booming of third-party SDKs allows the developer to create many interesting location-based AR applications. However, most users and application developers are unaware of the risk of potential location privacy leakage of their applications. Unlike smartphone where you can control when to turn on or off the sensors and applications, the mobile AR device continuously receives inputs from the environment through multiple sensors and the network. In this paper, we develop a novel user location tracking system – ARSpy, which could achieve accurate and involuntary tracking of the target by only monitoring the network throughput. Our real-world attack experiments on the Android platform show that our attack method achieves high localization accuracy and the attacker can recover the moving trajectory of the victim with high possibility. We have also proposed 3 mitigation mechanisms to mitigate such threats. Our study is expected to urge AR application developers to revise their geolocation transmission protocol and, more importantly, serve as a call for more attention from the application user and AR SDK designers to have the full knowledge of the potential risk brought by the location-based AR applications.

Acknowledgments

This work was supported in part by the NSF Grants CNS 1824440, CNS 1828363, CNS 1757533, CNS 1618398, CNS 1651947, and CNS 1564128.

References

[1] Battery historian, [Online]. Available: https://github.com/google/battery-historian
[2] My data manager.
[3] PokemonGo, [Online]. Available: https://www.pokemongo.com/
[4] Sift, [Online]. Available: https://github.com/agrinman/sift-ios
[5] WallaMe, [Online]. Available: http://walla.me
[6] World Brush, [Online]. Available: https://worldbrush.net
[7] N. Lomas, "Gatwick Airport now has 2,000 beacons for indoor navigation," [Online]. Available: https://techcrunch.com/2017/05/25/gatwick-airport-now-has-2000-beacons-for-i ndoor-navigation/
[8] X. Cai, X. C. Zhang, B. Joshi, and R. Johnson, "Touching from a distance: Website fingerprinting attacks and defenses," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 605–616.
[9] D. Chatzopoulos, C. Bermejo, Z. Huang, and P. Hui, "Mobile augmented reality survey: From where we are to where we go," *IEEE Access*, vol. 5, pp. 6917–6950, 2017.
[10] S. Chen, R. Wang, X. Wang, and K. Zhang, "Side-channel leaks in web applications: A reality today, a challenge tomorrow," in *Proc. IEEE Symp. Secur. Privacy*, 2010, pp. 191–206.
[11] A. Colceriu, 2016. [Online]. Available: https://www.ixiacom.com/company/blog/ catching-pokemon-go-your-network
[12] P. Dahne and J. N. Karigiannis, "Archeoguide: System architecture of a mobile outdoor augmented reality system," in *Proc. Int. Symp. Mixed Augmented Reality*, 2002, pp. 263–264.
[13] A. K. Das, P. H. Pathak, C.-N. Chuah, and P. Mohapatra, "Privacy-aware contextual localization using network traffic analysis," *Comput. Netw.*, vol. 118, pp. 24–36, 2017.
[14] J. A. de Guzman, K. Thilakarathna, and A. Seneviratne, "Security and privacy approaches in mixed reality: A literature survey," 2018, *arXiv: 1802.05797*.
[15] Y.-A. De Montjoye, C. A. Hidalgo, M. Verleysen, and V. D. Blondel, "Unique in the crowd: The privacy bounds of human mobility," *Sci. Reports*, vol. 3, 2013, Art. no. 1376.
[16] T. M. T. Do and D. Gatica-Perez, "Where and what: Using smartphones to predict next locations and applications in daily life," *Pervasive Mobile Comput.*, vol. 12, pp. 79–91, 2014.
[17] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Peek-a-boo, i still see you: Why efficient traffic analysis countermeasures fail," in *Proc. IEEE Symp. Secur. Privacy*, 2012, pp. 332–346.
[18] U. Fiore, F. Palmieri, A. Castiglione, V. Loia, and A. De Santis, "Multimedia-based battery drain attacks for Android devices," in *Proc. IEEE 11th Consum. Commun. Netw. Conf.*, 2014, pp. 145–150.
[19] P. Föckler, T. Zeidler, B. Brombach, E. Bruns, and O. Bimber, "PhoneGuide: Museum guidance supported by on-device object recognition on mobile phones," in *Proc. 4th Int. Conf. Mobile Ubiquitous Multimedia*, 2005, pp. 3–10.
[20] T. Höllerer, "User interfaces for mobile augmented reality systems," PhD thesis, Columbia Univ., New York, NY, 2004.
[21] T. Höllerer *et al.*, "User interface management techniques for collaborative mobile augmented reality," *Comput. Graph.*, vol. 25, no. 5, pp. 799–810, 2001.
[22] R. Hoyle, R. Templeman, S. Armes, D. Anthony, D. Crandall, and A. Kapadia, "Privacy behaviors of lifeloggers using wearable cameras," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.*, 2014, pp. 571–582.
[23] Y.-G. Kim and W.-J. Kim, "Implementation of augmented reality system for smartphone advertisements," *Int. J. Multimedia Ubiquitous Eng.*, vol. 9, no. 2, pp. 385–392, 2014.
[24] J. Krumm, "Inference attacks on location tracks," in *Proc. Int. Conf. Pervasive Comput.*, 2007, pp. 127–143.
[25] K. Lebeck, T. Kohno, and F. Roesner, "How to safely augment reality: Challenges and directions," in *Proc. 17th Int. Workshop Mobile Comput. Syst. Appl.*, 2016, pp. 45–50.
[26] K. Lebeck, K. Ruth, T. Kohno, and F. Roesner, "Securing augmented reality output," in *Proc. IEEE Symp. Secur. Privacy*, 2017, pp. 320–337.
[27] K. Lebeck, K. Ruth, T. Kohno, and F. Roesner, "Arya: Operating system support for securely augmenting reality," *IEEE Security Privacy*, vol. 16, no. 1, pp. 44–53, Jan./Feb. 2018.
[28] V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," *Soviet Phys. Doklady*, vol. 10, pp. 707–710, 1966.
[29] M. Li *et al.*, "All your location are belong to us: Breaking mobile social networks for automated user location tracking," in *Proc. 15th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, 2014, pp. 43–52.
[30] Y. Liang, Z. Cai, Q. Han, and Y. Li, "Location privacy leakage through sensory data," *Secur. Commun. Netw.*, vol. 2017, 2017, Art. no. 7576307.
[31] L. Lu, E.-C. Chang, and M. C. Chan, "Website fingerprinting and identification using ordered feature sequences," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2010, pp. 199–214.
[32] G. Meyer-Lee, J. Shang, and J. Wu, "Location-leaking through network traffic in mobileaugmented reality applications," in *Proc. IEEE 37th Int. Perform. Comput. Commun. Conf.*, 2018, pp. 1–8.
[33] B. Miller, L. Huang, A. D. Joseph, and J. D. Tygar, "I know why you went to the clinic: Risks and realization of HTTPS traffic analysis," in *Proc. Int. Symp. Privacy Enhancing Technol. Symp.*, 2014, pp. 143–163.
[34] A. Morrison *et al.*, "Like bees around the hive: A comparative study of a mobile augmented reality map," in *Proc. SIGCHI Conf. Hum. Factors Comput. Syst.*, 2009, pp. 1889–1898.
[35] M. Nasr, A. Houmansadr, and A. Mazumdar, "Compressive traffic analysis: A new paradigm for scalable traffic analysis," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 2053–2069.
[36] A. Panchenko *et al.*, "Website fingerprinting at internet scale," in *Proc. 23rd Internet Soc. Netw. Distrib. Syst. Secur. Symp.*, 2016.
[37] N. Raval, A. Srivastava, K. Lebeck, L. Cox, and A. Machanavajjhala, "Markit: Privacy markers for protecting visual secrets," in *Proc. ACM Int. Joint Conf. Pervasive Ubiquitous Comput.: Adjunct Publication*, 2014, pp. 1289–1295.
[38] P. Renevier and L. Nigay, "Mobile collaborative augmented reality: The augmented stroll," in *Proc. IFIP Int. Conf. Eng. Hum.-Comput. Interaction*, 2001, pp. 299–316.
[39] E. Rodrigues, R. Assunção, G. L. Pappa, D. Renno, and W. Meira Jr, "Exploring multiple evidence to infer users' location in Twitter," *Neurocomputing*, vol. 171, pp. 30–38, 2016.

[40] F. Roesner, T. Kohno, and D. Molnar, "Security and privacy for augmented reality systems," *Commun. ACM*, vol. 57, no. 4, pp. 88–96, 2014.

[41] F. Roesner, D. Molnar, A. Moshchuk, T. Kohno, and H. J. Wang, "World-driven access control for continuous sensing," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 1169–1181.

[42] R. Schuster, V. Shmatikov, and E. Tromer, "Beauty and the burst: Remote identification of encrypted video streams," in *Proc. 26th USENIX Conf. Secur. Symp.*, 2017, pp. 1357–1374.

[43] I. Sluganovic, M. Serbec, A. Derek, and I. Martinovic, "HoloPair: Securing shared augmented reality using microsoft hololens," in *Proc. 33rd Annu. Comput. Secur. Appl. Conf.*, 2017, pp. 250–261.

[44] D. Song, "Timing analysis of keystrokes and SSH timing attacks," in *Proc. 10th Conf. USENIX Secur. Symp.*, 2001, Art. no. 25.

[45] I. E. Sutherland, "A head-mounted three dimensional display," in *Proc. December 9–11, 1968, Fall Joint Comput. Conf., Part I*, 1968, pp. 757–764.

[46] R. Templeman, Z. Rahman, D. Crandall, and A. Kapadia, "PlaceRaider: Virtual theft in physical spaces with smartphones," 2012, *arXiv:1209.5982*.

[47] T. Tiwari, A. Klausner, M. Andreev, A. Trachtenberg, and A. Yerukhimovich, "Location leakage from network access patterns," in *Proc. IEEE Conf. Commun. Netw. Secur.*, 2019, pp. 214–222.

[48] C. V. Wright, L. Ballard, S. E. Coull, F. Monrose, and G. M. Masson, "Spot me if you can: Uncovering spoken phrases in encrypted VoIP conversations," in *Proc. IEEE Symp. Secur. Privacy*, 2008, pp. 35–49.

[49] C. V. Wright, S. E. Coull, and F. Monrose, "Traffic morphing: An efficient defense against statistical traffic analysis," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2009.

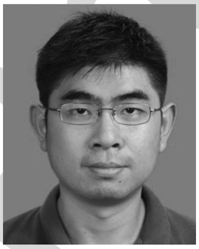
[50] H. Zang and J. Bolot, "Anonymization of location data does not work: A large-scale measurement study," in *Proc. 17th Annu. Int. Conf. Mobile Comput. Netw.*, 2011, pp. 145–156.

[51] Y. Zheng and X. Zhou, *Computing With Spatial Trajectories*. Berlin, Germany: Springer, 2011.

Jiacheng Shang (Student Member, IEEE) received the BEng degree in computer science from Nanjing University, China, in 2015. He is currently working toward the PhD degree in the Department of Computer and Information Sciences, Temple University, Philadelphia, Pennsylvania. His current research interests include the security and privacy issues in mobile cyber-physical systems.

Si Chen (Member, IEEE) received the PhD degree in computer science and engineering from the University at Buffalo–SUNY, Buffalo, New York, in 2016. He is currently an assistant professor with the Computer Science Department, West Chester University, West Chester, Pennsylvania. His research is supported by Microsoft. His current research interests include security and privacy in mobile sensing and cyberphysical systems. He was a recipient of the Best Student Paper Award of IEEE ICDCS 2017. He is a member of the ACM.

Jie Wu (Fellow, IEEE) is the director of the Center for Networked Computing and Laura H. Carnell professor at Temple University, Philadelphia, Pennsylvania. He also serves as the director of International Affairs at College of Science and Technology. He served as chair of Department of Computer and Information Sciences from the summer of 2009 to the summer of 2016 and associate vice provost for International Affairs from the fall of 2015 to the summer of 2017. Prior to joining Temple University, Philadelphia, Pennsylvania, he was a program director with the National Science Foundation and was a distinguished professor at Florida Atlantic University, Boca Raton, Florida. His current research interests include mobile computing and wireless networks, routing protocols, cloud and green computing, network trust and security, and social network applications. He regularly publishes in scholarly journals, conference proceedings, and books. He serves on several editorial boards, including the *IEEE Transactions on Mobile Computing*, the *IEEE Transactions on Service Computing*, the *Journal of Parallel and Distributed Computing*, and the *Journal of Computer Science and Technology*. He was general co-chair for IEEE MASS 2006, IEEE IPDPS 2008, IEEE ICDCS 2013, ACM MobiHoc 2014, ICPP 2016, and IEEE CNS 2016, as well as program co-chair for IEEE INFOCOM 2011 and CCF CNCC 2013. He was an IEEE Computer Society distinguished visitor, ACM distinguished speaker, and chair for the IEEE Technical Committee on Distributed Processing (TCDP). He is a fellow of the AAAS. He is the recipient of the 2011 China Computer Federation (CCF) Overseas Outstanding Achievement Award.



Shu Yin (Member, IEEE) received the BS degree in communication engineering from the Wuhan University of Technology (WUT), China, in 2006 the MS degree in signal and information processing from the Wuhan University of Technology, China, in 2008, and the PhD degree in computer science from Auburn University, Auburn, Alabama, in 2012. Currently, he is an assistant professor/ research scientist with the School of Information Science and Technology, ShanghaiTech University, China. Prior to joining ShanghaiTech University, China, in 2016, he had been an assistant professor and associate professor with the College of Computer Science and Electronics Engineering, Hunan University, China, for 4.5 years. He has worked as a post-doctoral researcher at the National University of Defense Technology, China from 2014 to 2017. His research interests include storage systems, reliability modeling, fault tolerance, energy-efficient computing, high performance computing, and wireless communications.