5-2013

Anonymous authentication of visitors for mobile crowd sensing at amusement parks

Divyan KONIDALA
Singapore Management University, divyanmk@smu.edu.sg

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

Yingjiu LI
Singapore Management University, yjli@smu.edu.sg

Hoong Chuin LAU
Singapore Management University, hclau@smu.edu.sg

Stephen FIENBERG
Carnegie Mellon University

DOI: https://doi.org/10.1007/978-3-642-38033-4_13

Published in Information Security Practice and Experience. ISPEC 2013. Lecture Notes in Computer Science, vol 7863. Springer, Berlin. pp. 174-188.
http://doi.org/10.1007/978-3-642-38033-4_13

Anonymous Authentication of Visitors for Mobile Crowd Sensing at Amusement Parks

Divyan Munirathnam Konidala[1], Robert H. Deng[1], Yingjiu Li[1], Hoong Chuin Lau[1], and Stephen E. Fienberg[2]

[1] School of Information Systems, Singapore Management University, 80 Stamford Road, Singapore 178902
{divyanmk,robertdeng,yjli,hclau}@smu.edu.sg

[2] Department of Statistics, Machin Learning Department, Heinz College, and Cylab Carnegie Mellon University, Pittsburgh, PA, 15213-3890 USA
fienberg@stat.cmu.edu

Abstract. In this paper we focus on authentication and privacy aspects of an application scenario that utilizes mobile crowd sensing for the benefit of amusement park operators and their visitors. The scenario involves a mobile app that gathers visitors' demographic details, preferences, and current location coordinates, and sends them to the park's sever for various analyses. These analyses assist the park operators to efficiently deploy their resources, estimate waiting times and queue lengths, and understand the behavior of individual visitors and groups. The app server also offers visitors optimal recommendations on routes and attractions for an improved dynamic experience and minimized wait times. We propose a practical usable solution we call an anonymous authentication of visitors protocol that protects the privacy of visitors even while collecting their details, preferences and location coordinates; deters adversaries outside the park from sending in huge amounts of false data, which lead to erroneous analyses and recommendations and bring down the app server. We utilize queuing theory to analyze the performance of a typical app server receiving numerous simultaneous requests from visitors to process a core function of our protocol.

Keywords: Mobile crowd sensing, Amusement park, Anonymous authentication, False data, Partially blind signature scheme.

1 Introduction

Smart mobile devices allow users to download, install, and run mobile (software) applications (apps) that allow users to receive locations based services. For example, mobile apps use the Global Positioning System (GPS) sensor extensively to provide information specific to the users' current location. For mobile crowd sensing [12], [20], a mobile app periodically collects a device's sensor data from a large group of people and transmits them to the app server. Analyses at the server can provide location specific information directly to individuals and groups.

Amusement and theme parks, such as Disneyland, Universal Studios, and LEGOLAND, offer a variety of attractions, including rides, shows, dining, and other forms of entertainment. Since they attract a large number of visitors, the major inconvenience faced by visitors is long wait times, and park operators are always exploring and implementing ways and means to minimize/control queue lengths and enhance visitor experiences.

1.1 Current Situation

Park operators have several options to minimize visitors' wait times and to improve their overall experience. In the following, we briefly describe three most common approaches, where visitors' involvement is required.

Special/Express Pass Approach. Parks can sell special admission tickets, at a premium which allow purchasers to bypass the regular lines and gain priority entrance [22]. This approach may create a feeling of frustration when the special pass holders bypass those visitors who are waiting in the regular lines for a long time.

Timed Ticket Approach. The visitor inserts his/her admission ticket into a machine (located near the attraction) that issues a timed ticket with the return time window printed on it [9]. Now the visitor is free to enjoy the rest of the park and need to reach the attraction within the return time window. Since visitors cannot purchase such timed tickets, this approach gives any visitor inside the park equal opportunity to pick up a timed ticket. Typically timed tickets are issued in limited number; visitors must be able to physically reach the machine before they are all issued; otherwise they need wait in the regular lines. At any point of time, a visitor cannot possess more than one valid timed ticket and if the visitor fails to reach the attraction around his/her return time window, the timed ticket will be wasted. As a result this approach is neither scalable, flexible nor dynamic with respect to the total number of visitors and their movements in the park.

Mobile App Approach. Some park operators deploy mobile apps [23], [18], which display the map of the park, offer directions, provide information about promotions, shopping and dining options, and most importantly information about various attractions and their wait times. Such apps push notifications from the app server to the visitors' smart mobile devices. They do not generally aggregate the mobile crowd sensed data and visitors' personal preferences, nor analyze them in order to recommend optimal routes and attractions, for an enhanced dynamic visitor experience and to minimize wait times.

1.2 Application Scenario

Building upon the mobile app approach, we envision an application scenario, which is based on the mobile crowd sensing and would eliminate the drawbacks of the previously described approaches and offers other great benefits to both the park operators and their visitors.

Visitor Alice downloads, installs and runs a mobile app developed by the park operator. The mobile app prompts Alice to enter demographic details such as her age, gender, nationality, height (used to determine access to certain attractions), dietary restrictions and other health issues, as well as preferences for rides; must go and must skip attractions, etc. The app would not ask for Personal Identifying Information (PII) such as name, social security number, and address. The app then sends these details and preferences to the app server managed by the park operator. Henceforth, the app would also periodically collect Alice's current GPS location coordinates and send them to the app server, and Alice would receive communications from the server.

Benefits to the Visitors. Based on Alice's current location and her demographic details and preferences, the server would periodically and dynamically calculate an updated personalized itinerary for Alice to visit various attractions that minimize her wait times. Alice can follow the itinerary to visit an attraction during the recommended time slot, and tap her smart device or an RFID enabled device at the entrance/exit of the attraction for validation. Incentives (such as points that can be redeemed for gifts) can be used to encourage Alice to follow the recommended route.

With this approach, visitors need not physically visit a machine to obtain a return timed ticket. All visitors using the mobile app have equal opportunity to receive personalized time slots and recommended routes. If the server analyzes that a particular visitor, based on his/her current location cannot reach the attraction within his/her recommended time slot, the server would dynamically recalibrate new time slots for that visitor. Therefore this scenario is scalable, dynamic, and certainly not wastage prone.

Benefits to the Park Operators. By dynamically analyzing the visitors' crowd-sensed GPS data, the app server assists the park operators to manage and deploy their resources efficiently, manage traffic flows and congestion, analyze various key performance indices such as the queue information, and average/maximum wait times at each attraction, etc. Similarly by analyzing visitors' demographic details, preferences and activities, the app server also assists park operators to gain insight into the behavior of groups and individual visitors based common preferences, and background characteristics. This then allows for new approaches to meet visitor needs as well as dynamic optimal recommendations and routes to improve their overall experience in the park.

2 Threats and Security Requirements

From our application scenario, we identified the following threats to privacy and certain security requirements to alleviate these threats.

2.1 Threats

Visitor Privacy Violation. In our application scenario the mobile app gathers visitors' demographic details, preferences and current location coordinates;

therefore, we have to make sure that all these details do not reveal and cannot be linked to the true identities of the visitors; otherwise the app server or a hacker who has hacked into the app server, can generate detailed profiles of the visitors, their buying interests, track all their activities and current locations and carry out malicious acts such as identity fraud, stalking, and sending spam adverts.

False Data. Adversaries outside the park, for various malicious reasons (extortion, blackmail) can attempt to emulate the mobile app in a computer and create an unlimited number of fake virtual visitors with their location coordinates inside the park. The app sever might unsuspectingly consider all these fake virtual visitors to be actual visitors inside the park. Sending huge amounts of such false data to the app server would result in erroneous analyses and recommendations that could confuse and frustrate the visitors and also overwhelm, degrade and eventually bring down the server.

Greedy Visitors. Greedy park visitors could attempt to tamper with their smart mobile devices [15], to send false location coordinates in order to cheat the process, obtain unfair preferential treatment, rewards, and earlier time slots for the attractions of their choice.

Man-In-Middle-Attacks. The channel between the mobile app and the app server is potentially prone to eavesdropping, data capture, and data corruption by hackers. Since the channel is carrying potentially sensitive data, such attacks would violate visitors' privacy and lead to erroneous analyses at server.

2.2 Security Requirements

Use of Pseudonyms. Visitors must interact with the app server using pseudonyms to decouple visitors' data from their true identities.

Visitor Authentication. To prevent adversaries outside the park from supplying large volume of false data, the app server needs to verify whether the data it receives is indeed from a visitor inside the park. To accomplish this, the app server must authenticate the visitors inside the park and receive data from only such authenticated visitors.

Data Auditing. The app server needs to audit the data it receives in order to detect any anomalies or false data from greedy visitors.

Secure Communication Channel. The channel between the mobile app and the app server must be secure enough to provide app server authentication to the mobile app, and data protection and integrity for communications.

3 Proposed Anonymous Authentication of Visitors (AAV) Protocol

Nothing would stop an adversary outside the park from supplying a large volume of false data. Therefore, we need to authenticate the information reported by

a visitor inside the park without divulging the visitor' identity. In this paper we apply the cryptographic notion of anonymous authentication: authenticating the visitor without revealing his/her identity. The fundamental idea here is to interact with the visitor using a pseudonym instead of his/her true identity.

3.1 Naive Approaches

One naive approach would be to use the admission ticket ID as the visitor's pseudonym. The mobile app would send to the app server, the visitors' demographic details, preferences, and current location coordinates referring the ticket ID. Linking these details to the ticket ID proves that the visitor is genuine and is indeed inside the park. The adversary must purchase a sufficiently large number of valid admission tickets to launch a successful attack, but this would not be economical as the tickets are very expensive. This naive approach would greatly limit the adversary's power; however it does not protect the privacy of the visitor. A vast majority of the visitors buy their tickets using their credit cards. In which case the issued ticket IDs are recorded and linked to the credit card, which is in turn linked to the true identity of the visitor. Therefore, this naive approach does not truly address the requirement of anonymous authentication.

A second naive approach is for the app server to accept communications that come only through the Wi-Fi network of the park. Once again here, without authentication, nothing would prevent the adversary to use the park's Wi-Fi network to create unlimited number of fake virtual visitors. Furthermore, most of the visitors may hesitate using an unsecured open Wi-Fi network and instead prefer to use their own 3G/4G network.

3.2 Background

Our proposed AAV protocol utilizes pseudonyms and partially blind signature scheme.

Blind Signature. In 1982, David Chaum proposed a new cryptographic primitive called the blind signature [5], which could be used as a primer tool to design electronic payment and electronic voting schemes with user privacy-protection in mind. Blind signature is a special kind of digital signature [17], which allows users to get signatures on their messages from authorized entities/signature issuers (e.g. banks, trusted third parties) without revealing the message contents to the authorized entity. Furthermore, if malicious signature issuers and verifiers (e.g. service providers, merchants) collude, they cannot discover the real identity of the user who actually holds the signatures.

Partially Blind Signature. Blind signatures provide total privacy for users by fully hiding messages (to be signed) from the signer. This property is not desired from the signer's point of view, because he is responsible for his signatures and he needs to know what he would be signing on. To achieve a solution acceptable for both the signer and users, Abe and Fujisaki proposed the idea of partially

blind signature [1], which was later formally proved by Abe and Okamato [2]. A partially blind signature scheme has two portions: one portion consists of the message that is hidden by the user (as in blind signature scheme) and in the other portion, the signer can explicitly embed necessary information such as issuing date, expiry date, signer's identity etc. Users should be made aware of the information that the signer wishes to embed into the signature. Users must also be able to verify that only the agreed-upon information has been embedded by the signer; otherwise the signer may secretly embed undisclosed information into the signature that could reveal the true identity of the users at a later stage.

3.3 AAV Protocol Description

The AAV protocol is executed in two phases: "Certified Pseudonym Issuing Phase", followed by the "Subsequent Interaction Phase".

In the "Certified Pseudonym Issuing Phase", Alice's mobile app generates a pseudonym P and utilizes the partially blind signature scheme to hide P in a blinded message B. The mobile app then sends the ticket ID along with B to the app server. The app server verifies the validity of ticket ID, and inputs an expiry date while digitally signing B. As a result the app server has no clue about Alice's pseudonym and cannot link the future communications from this pseudonym to Alice. The mobile app would unblind the signature on B, in order to recover the signature S to the bare pseudonym P, thus making S the certified pseudonym by the app server.

In the "Subsequent Interaction Phase", the mobile app no longer uses the ticket ID, instead it uses the P and the S to send Alice's demographic details, preferences, current location coordinates. Since the signature on B from the app server has been unblinded to the bare pseudonym P, the app server can easily verify whether the pseudonym P sent by the mobile app matches with the pseudonym signed in the signature S and also whether it is within the expiry date. As a result the app server can make sure that it is communicating with a certified pseudonym/visitor inside the park.

Setup. We construct our protocol using the RSA-based partially blind signature scheme proposed by Abe and Fujisaki [1]. The mobile app and the app server share a secure one-way hash function $h(.)$ whose length is k bits. The app server executes RSA function as follows: N is a product of two large primes p and q. N satisfies $S_i \nmid \lambda$ for all prime $S_i (3 \leq S_i \leq 2^k - 1)$, where λ is the LCM of $(p-1)$ and $(q-1)$. The prime e is an RSA public component, which is larger than or equal to $2^k - 1$. The corresponding private key is d given by $ed = 1 mod \lambda$. The mobile app has the knowledge of e and N.

It is a known fact that a one day admission ticket would expire by the end of that day the visitor enters the park and a two day admission ticket would expire by the end of the second day of the visitor's visit to the park. Therefore, we assume that both the mobile app and the app server have the knowledge of the expiry date of an admission ticket. Let x be the expiry date of the admission ticket, whose length is $k-2$ bits and both the mobile app and the app server are

capable of calculating: $\tau(x) = 2^{k-1} + 2h(x) + 1$. $\tau(x)$ is a formatting function designed to keep its domain in $2^{k-1} < \tau(x) < 2^k$ so that $\tau(x_i)$ does not divide $\tau(x_j)$ where $i \neq j$. Also, it is designed to produce odd numbers only so that it becomes relatively prime with λ.

The mobile app can also generate pseudonyms of length k bits and the communication channel between the mobile app and the app server is secured via the standard HTTPS (Hypertext Transfer Protocol Secure) protocol [16].

Certified Pseudonym Issuing Phase depicted in the Fig.1 is self explanatory; however, we elaborate some of the steps here. Step 3 executes the blinding procedure of the partially blind signature scheme, which hides P in B. No one else other than the mobile app knows the value of P, i.e., the blinded message B is statistically or perfectly indistinguishable from P as long as blinding factor R is not revealed. Step 6 validates whether *tktid* is a valid unused ticket and has not been previously used to generate a partially blind signature. In Step 3, $e\tau(x)$ has become the public key that contains the common information between the mobile app and the app server, i.e., the expiry date. Therefore, in step 7, the app server calculates the corresponding private key d_x. Step 8 executes the

Alice's Mobile App	Secure Channel HTTPS	Park's App Server
Hash function: $h(.)$		Hash function: $h(.)$
		Large primes: p, q; $N = pq$
		$\lambda = \text{LCM}(p-1, q-1)$
		$ed = 1 mod \lambda$
Server's public key: e, N		Private key: d, N
Ticket's expiry date: x		Ticket's expiry date: x
$\tau(x) = 2^{k-1} + 2h(x) + 1$		$\tau(x) = 2^{k-1} + 2h(x) + 1$
1. Scan ticket barcode: *tktid*		
2. Generate pseudonym: P		
3. Randomly choose a blind factor: $R \in Z_N^*$		
4. blind(P): $B = h(P)R^{e\tau(x)} mod N$		
	5. $(tktid, B) \longrightarrow$	
		6. Validate: *tktid*
		7. Calculate private key: $d_x = 1/e\tau(x) mod \lambda(N)$
		8. ParBlindSign(B): $\Phi = B^{d_x} mod N$
	$\longleftarrow$ 9. (Φ)	
10. unblind(Φ): $S = \Phi/R \equiv P^{d_x} mod N$		
11. Verify: S using $e\tau(x)$		

Fig. 1. Certified Pseudonym Issuing Phase of AAV Protocol

signing procedure of the partially blind signature scheme to generate the blinded signature Φ. Step 10 executes the unblinding procedure of the partially blind signature scheme on Φ, which unblinds B to reveal P in the signature S. From here on S certifies the pseudonym P.

Subsequent Interaction Phase (Fig.2). In this phase, the mobile app sends the visitor's demographic details and preferences ($DetPre$), and current GPS coordinates (Gps) using the S, P, x. The app server computes $h(P)$ and verifies the signature on S using $x, h(P), d_x$, this validates that P has been indeed certified in S. With P being the reference index in the database, the app server records and analyzes the $DetPre$, and the periodic ($Time : Gps$) data, and $Rewards$ calculations. The server would now keep track and communicate with the mobile app using this P. The mobile app would receive the optimal route ($Route$), and the dynamically calibrated personalized time slots ($TSlots$) for various attractions ($Attrs$) in the park, as well as $Rewards$.

Alice's Mobile App	Secure Channel HTTPS	Park's App Server
1. Enter demographic details and preferences: $DetPre$ 2. Obtain current GPS location: Gps 3. Retrieve: $\{S, P, x\}$		
	$\xrightarrow{4.\ (\{S,P,x\}, DetPre, Gps)}$	
		5. Verify: S using $x, h(P), d_x$ and validate: P 6. Store and analyze: $p : DetPre$ $p : (Time_1 : Gps_1, \cdots, Time_n : Gps_n)$ 7. Compute and Store: $p : Route$ $p : (Attr_1 : TSlot_1, \cdots, Attr_n : TSlot_n)$ $p : Rewards$
	$\xleftarrow{8.\ (Route, Attrs : TSlots, Rewards)}$	
9. View: Optimal $Route$ $Attr_1 : TSlot_1, \cdots, Attr_n : TSlot_n$ $Rewards$		

Fig. 2. Subsequent Interaction Phase of AAV Protocol

3.4 Anonymous Authentication of a Group

In our application scenario, the app server would assist the park operators to understand the behavior of groups moving together by constantly analyzing

the visitors' demographic details, preferences and activities; however, our AAV protocol is only applicable to individual visitors. Therefore, we extend the AAV protocol to accommodate anonymous authentication of a group. The basic idea here is that the pseudonyms of individual members of a group would all be linked to a single common group pseudonym. With this approach the app server can carry out behavioral analysis of individual members of a group based on their individual unique pseudonyms and also the behavioral analysis of the entire group based on their single common group pseudonym.

We can slightly modify the "Certified Pseudonym Issuing Phase", so that the head/leader of the group's mobile app would generate two pseudonyms; one representing the group pseudonym (GP) and the second representing his/her own individual pseudonym (P). Both the GP and P, would be hidden in the blinded message B. The head of the group would then inform the rest of the group members about the group pseudonym, e.g., via email. The rest of the group members' mobile apps would then include this GP along with their individual pseudonyms during their "Certified Pseudonym Issuing Phase". Finally, during the "Subsequent Interaction Phase", the app server would record both the group pseudonym and the individual pseudonyms.

4 Security Analysis

This section provides security analysis of our AAV protocol with respect to the threats described in section 2.

4.1 Use of Pseudonyms to Protect Visitors' Privacy

Our proposed AAV protocol successfully utilizes pseudonyms to decouple visitors' data from their true identities. The app server has no role in generating the pseudonyms for the visitors. The blinding procedure of the partially blind signature scheme does not reveal the visitor's pseudonym to the app server, yet the scheme allows us to obtain the signature of the app server on the pseudonym. The app server cannot link the pseudonym to neither the ticket ID nor the credit card used to purchase the ticket. Both the mobile app and the app server can independently produce $x, \tau(x)$, and $e\tau(x)$; therefore the mobile app can precisely verify (step 11 of the "Certified Pseudonym Issuing Phase") that apart from the expiry date x, the app server has not included any hidden message to distinguish the transaction later.

Restricted Privacy. Our AAV protocol provides only restricted privacy, but not complete visitor anonymity and unlinkability. Our application scenario requires that the visitor be tracked with a particular pseudonym, so that his/her preferences and current GPS location data can be gathered, analysed, and used to recommend optimal route, award rewards, and send dynamically calibrated personalized time windows for various attractions in the park.

Physical Layer Anonymity. Even though we use the AAV protocol, visitors may be tracked based on their smart device's MAC (Media Access Control) address, which is a unique fixed identifier assigned to network interfaces for communications on the physical network. However, if the visitor is using the device's 3G/4G network to communicate with the app server, the operator of the 3G/4G network would assign a different IP address each time a connection is made and the MAC address is made known only to the operator. The app server would only know the dynamic IP address, which cannot be used to track the device.

On the other hand if the visitor is using the free Wi-Fi network provided by the park operator, there are chances that the app server may retrieve and store the MAC addresses off the Wi-Fi access points. This situation is very rare and would require considerable amount of resources on the part of the app server to record the MAC address of every communication. However, there are ways to circumvent this problem; the visitor may choose to communicate with the app server through anonymity networks like the mix network [11], and Tor onion routing network [21]. Such networks direct user's internet traffic through a worldwide volunteer network of servers to conceal a user's location or guard against network surveillance or traffic analysis. There exists an open source client for the Tor network on Android mobile devices called the "Orbot" [19].

4.2 Visitor Authentication to Deter False Data from Adversaries Outside the Park

Our AAV protocol achieves anonymous authentication, whereby the app server accepts data only from pseudonyms that have been certified during the "Certified Pseudonym Issuing Phase". The function $\tau(x)$ prevents a visitor to obtain multiple signatures on the same pseudonym with different expiry dates [1]. The $h(P)$ in the step 4: blind(P): $B = h(P)R^{e\tau(x)} mod N$, prevents two visitors with valid certified pseudonyms to collude and forge a new valid certified pseudonym.

Uniqueness of Ticket ID. In our "Certified Pseudonym Issuing Phase" we depend on the ticket ID to be unique and non-sequential. The app server verifies whether the ticket ID is un-used, un-expired, and was not previously used to obtain the partially blind signature. But, if the park operators do not issue unique and non-sequential tickets, any one could produce and sell fake tickets and can also misuse our protocol. It is a problematic situation for both the park operators and for our protocol. In cases where the ticket ID is already on the ticket and not uniquely generated and printed at the time of issuing, we can expect sequential IDs. We suggest that the park operators generate and print another unique number (for example the current date and time) on every ticket at the time of issuing. In such a scenario, the mobile app would prompt the visitor to type in that unique number and send it along with the $(tktid, B)$, step 5 of "Certified Pseudonym Issuing Phase". This approach would ensure that we are dealing with unique and non-sequential ticket IDs.

4.3 Data Auditing: Heuristics, Thresholds, and Revocation to Deter Greedy Visitors

Following the approach of [15], which used heuristics to detect fake-location attacks against location-based services, we suggest that the app server would formulate and put in place certain heuristics such as calculating the time elapsed between the visitors' previous location and the current location. If this time matches with the average time taken by other visitors to commute between the same two locations, then the data is considered legitimate. Minimum threshold values must also be put in place, to detect false data. Whenever the app server identifies a particular pseudonym sending in data that does not match these heuristics and threshold values, it could be a greedy visitor, in which case the app server can immediately black list that particular certified pseudonym and deny all future communications.

4.4 Secure Channel to Counter Man-in-the-middle Attacks

In our AAV protocol, the communication channel between the mobile app and app server is secured using the standard HTTPS protocol [16]. The HTTPS authenticates the app server, and guarantees confidentiality and integrity for the data communicated between the mobile app and app server. The developers of the mobile app must carefully implement the HTTPS protocol; recent works [10], [13], have shown that improper implementations and over looking of various critical settings of HTTPS have resulted in complete breakdown of certificate verification, which can lead to successful man-in-the-middle attacks.

5 Related Work

There exist other cryptographic solutions such as the group signature schemes [7], and anonymous credential schemes [6], [8]. These schemes allow a member of a group to sign on a message on behalf of the group. The verifier of the signed message can prove that the message has come from the group, but cannot deduce the true identity of the group member who signed the message. At the outset these schemes seem suitable, but for the following reasons they are not practical for our application scenario. These schemes require all the visitors in the park to have a public and private key pair. They also require a group manager to add members into the group, issue group public key and to certify the group members' credentials. The group manager, if the need arises, can also trace and identify the member who signed a particular message. In our application scenario, the visitors enter the park in huge numbers in an unpredictable manner; we cannot expect the visitors to generate public and private key pairs; it is impossible for the visitors to establish a group manager among themselves and to execute complex operations of these signature schemes. The park operator cannot be a group manager, in which case the true identities of the visitors are revealed.

There also exist anonymous e-token schemes [4], where an user is initially issued a certain number of certified pseudonyms or anonymous e-tokens by the

server. At a later phase, every time the user communicates with the server, he/she uses a different anonymous e-token. This scheme does not reveal the true identity of the user, yet the server can confirm if the data has come from an authorized user possessing certified anonymous e-token. As mentioned in our security analysis section—Restricted Privacy—this scheme is also not practical for our application scenario because it provides complete anonymity and unlinkability of pseudonyms.

6 App Server Performance Results

In our proposed AAV protocol, the app server executes two core cryptographic procedures; the "partially blind signing procedure" (during the Certified Pseudonym Issuing Phase) and the "signature verification procedure" (during the Subsequent Communication Phase). Similar to the typical RSA-based signing and signature verification procedures [17]; both the RSA-based partially blind signing and signature verification procedures [1] include one hash operation and one exponentiation operation. We relied on [24] for the speed benchmarks of some of the most commonly used cryptographic algorithms. A 1024 bit RSA-based signing procedure and signature verification procedures take 0.67 milliseconds (ms) and 0.04 ms respectively, when executed on an AMD Opteron 8354, 2.2 GHz processor under Linux. We can assume that a 1024 RSA-based partially blind signing procedure and signature verification procedures would not take more than 0.67 ms and 0.04 ms respectively, when executed on an AMD Opteron 8354, 2.2 GHz processor under Linux.

Amusement parks attract large numbers of visitors. For example, the Magic Kingdom at Walt Disney World Resort, Florida, USA is the largest amusement park worldwide in order of annual attendance [3]; 17 million visitors in the year 2011, averaging 46,000 visitors a day. As a result, thousands of visitors' mobile apps would concurrently access the app server. Since a 1024 bit RSA-based partially blind signing procedure requires longer time (0.67 ms), when compared to the signature verification procedure (0.04 ms), we are particularly interested in the number of simultaneous partially blind signing requests that could be handled by an app server. Therefore we applied queuing theory [14] to analyze the performance of the app server by predicting its response times while executing the 1024 bit RSA-based partially blind signing procedure.

In queuing theory, a system consists of a single queue of jobs submitted to one or more servers. We used the stochastic $M/M/C$ queuing model, where the M represents the Markov or memory less or exponential nature of the job arrival and job service rates, and C is the number of servers attached to the queue. Initially we considered 1 app server, i.e., $M/M/1$ queuing model. A single server can process approximately 1,492 partially blind signing requests per second at the rate of 0.67 ms per request. We subjected this $M/M/1$ queuing model with increasing number of requests per second and calculated the respective server's average response time. The average response time is the sum of the average amount of time that it takes a server to process such a request, and the average amount of time a request spends in the queue.

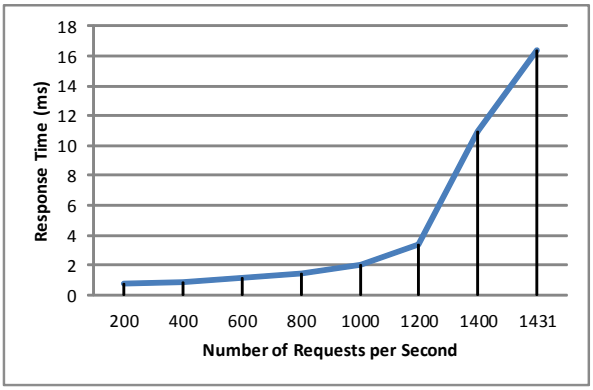

Fig. 3. Average Response Time of App Server with $M/M/1$ Queuing Model

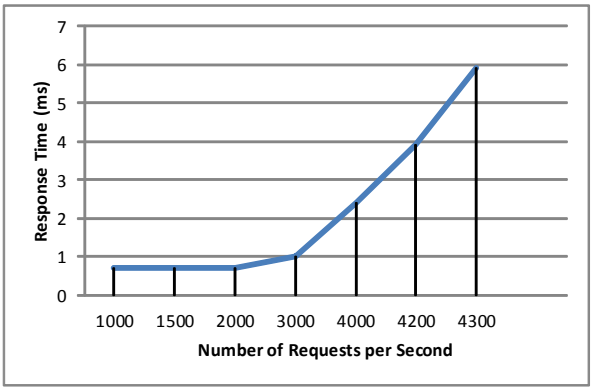

Fig. 4. Average Response Time of App Server with $M/M/3$ Queuing Model

Fig.3 depicts the chart of server's average response time with respect to the increasing number of requests per second. It shows that as the number of requests per second gets close to 1,492, the app sever becomes unstable, exponentially reaching: maximum utilization capacity and maximum number of requests that can be processed and are waiting in the queue, leading to a dead lock, and finally the average response time reaching one second per request, frustrating the visitors of the park. Fig.3 also points to the fact that the arrival rate of 1,000 requests per second allows the app server to be stable and process all the requests efficiently with an average response time of just 2 ms. However, considering the huge number of visitors in the park, we estimate that at least 3,000 (partially blind signing) requests per second must be processed by the app server. Therefore, we must increase the number of app servers, i.e., the value of

C in the $M/M/C$ queuing model. Our calculations as depicted via the chart in Fig.4 show that the $M/M/3$ queuing model, where a single queue of requests is now handled by 3 app servers, can efficiently process 3,000 requests per second, with an average response time of just 1 ms.

7 Conclusion

In this paper, we analyzed the authentication and privacy aspects of an application scenario that utilizes mobile crowd sensing for the benefit of amusement park operators and their visitors. We proposed a simple and practical anonymous authentication of visitors protocol that utilizes pseudonyms and partially blind signature scheme. The protocol protects the privacy of the visitors while authenticating them to the park's server and also prevents adversaries outside the park from bombarding the server with huge amounts of false data. We have utilized $M/M/C$ queuing model to analyze the server performance while receiving a large number of simultaneous partially blind signing requests (per second) from the visitors and recommended a minimum of 3 servers to handle 3,000 such requests per second for an optimal server response time. We offered several security discussions that need to be considered while deploying the application scenario. In fact, the contributions of this paper would be applicable to other types of similar application scenarios that are based on mobile crowd sensing and incentivizing the visitors.

Acknowledgement. This research/project is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office.

References

1. Abe, M., Fujisaki, E.: How to date blind signatures. In: Kim, K., Matsumoto, T. (eds.) ASIACRYPT 1996. LNCS, vol. 1163, pp. 244–251. Springer, Heidelberg (1996)
2. Abe, M., Okamoto, T.: Provably secure partially blind signatures. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 271–286. Springer, Heidelberg (2000)
3. AECOM, TEA-AECOM 2011 Theme Index The Global Attractions Attendance Report, Themed Entertainment Association (TEA) (2011)
4. Camenisch, J., Hohenberger, S., Kohlweiss, M., Lysyanskaya, A., Meyerovich, M.: How to win the clone wars: Efficient periodic n-times anonymous authentication. In: CCS 2006, pp. 201–210 (2006)
5. Chaum, D.: Blind signatures for untraceable payments. In: CRYPTO 1982, pp. 199–203 (1982)
6. Chaum, D.: Security without identification: Transaction systems to make big brother obsolete. Communications of the ACM 28(10), 1030–1044 (1985)
7. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)

8. Damgård, I.B.: Payment systems and credential mechanisms with provable security against abuse by individuals. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 328–335. Springer, Heidelberg (1990)
9. Disney's FASTPASS Service, http://disneyworld.disney.go.com/guest-services/fast-pass/
10. Fahl, S., Harbach, M., Muders, T., Smith, M., Baumgartner, L., Freisleben, B.: Why Eve and Mallory love Android: An analysis of Android SSL (In)security. In: CCS 2012, pp. 50–61 (2012)
11. Berthold, O., Federrath, H., Köpsell, S.: Web mixes: A system for anonymous and unobservable internet access. In: Federrath, H. (ed.) Anonymity 2000. LNCS, vol. 2009, pp. 115–129. Springer, Heidelberg (2001)
12. Ganti, R., Ye, F., Lei, H.: Mobile crowdsensing: Current state and future challenges. IEEE Communications Magazine 49(11), 32–39 (2011)
13. Georgiev, M., Iyengar, S., Jana, S., Anubhai, R., Boneh, D., Shmatikov, V.: The most dangerous code in the world: Validating SSL certificates in non-browser software. In: CCS 2012, pp. 38–49 (2012)
14. Gross, D., Shortle, J.F., Thompson, J.M., Harris, C.M.: Fundamentals of Queueing Theory. Wiley (2008)
15. He, W., Liu, X., Ren, M.: Location cheating: A security challenge to location-based social network services. In: ICDCS 2011, pp. 740–749 (2011)
16. Internet Engineering Task Force (IETF), Network Working Group, HTTP Over TLS, RFC2818 (2000), http://tools.ietf.org/html/rfc2818
17. Menezes, A.J., vaz Oorschot, P.C., Vanstone, S.A.: Digital Signatures. In: Handbook of Applied Cryptography, ch.11. CRC Press (1997)
18. Merlin Entertainments iTunes App., LEGOLAND California (2012), https://itunes.apple.com/us/app/legoland-california-official/id452395530
19. Orbot: Tor on Android, The Tor Project (2012), https://guardianproject.info/apps/orbot/
20. Sherchan, W., Jayaraman, P.P., Krishnaswamy, S., Zaslavsky, A.B., Loke, S.W., Sinha, A.: Using on-the-move mining for Mobile crowdsensing. In: MDM 2012, pp. 115–124 (2012)
21. Tor, Anonymity Online, https://www.torproject.org/
22. Universal Express Passes, Universal Orlando Resort, http://www.universalorlando.com/Theme-Park-Tickets/Universal-Express/Express-Passes.aspx
23. Walt Disney iTunes App., Disney Mobile Magic (2012), https://itunes.apple.com/us/app/disney-mobile-magic/id500000336
24. Dai, W.: Speed Comparison of Popular Crypto Algorithms, http://www.cryptopp.com/benchmarks.html